# NAVAL POSTGRADUATE SCHOOL
## Monterey, California


# THESIS

**SNMP OVER WI-FI WIRELESS NETWORK**

by

Jiradett Kerdsri

March 2003

| | |
|---|---|
| Thesis Advisor: | Ted Lewis |
| First Reader: | Geoffrey Xie |
| Second Reader: | Gurminder Singh |

THIS PAGE INTENTIONALLY LEFT BLANK

| REPORT DOCUMENTATION PAGE | | | Form Approved OMB No. 0704-0188 |
|---|---|---|---|
| Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503. | | | |
| **1. AGENCY USE ONLY** *(Leave blank)* | **2. REPORT DATE** March 2003 | **3. REPORT TYPE AND DATES COVERED** Master's Thesis | |
| **4. TITLE AND SUBTITLE**: SNMP Over Wi-Fi Wireless Networks | | | **5. FUNDING NUMBERS** |
| **6. AUTHOR(S) Jiradett Kerdsri** | | | |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Naval Postgraduate School Monterey, CA 93943-5000 | | | **8. PERFORMING ORGANIZATION REPORT NUMBER** |
| **9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)** N/A | | | **10. SPONSORING/MONITORING AGENCY REPORT NUMBER** |
| **11. SUPPLEMENTARY NOTES** The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. | | | |
| **12a. DISTRIBUTION / AVAILABILITY STATEMENT** Approved for public release; distribution is unlimited | | | **12b. DISTRIBUTION CODE** |

**13. ABSTRACT** ***(maximum 200 words)***


Simple Network Management Protocol (SNMP) allows users of network equipment (i.e. Network Administrators) to remotely query the state of any device being tested for system load, utilization and configuration.

Windows NT, Windows 2000 and Windows XP Professional are all equipped with SNMP service so that an SNMP manager can communicate with an SNMP agent running on a wireless 802.11b client. However the rest of Windows operating systems, including Windows CE and a Pocket PC, have to run third party proxy SNMP agents in order to be recognized by an SNMP management application.

This thesis describes an implementation of a Pocket PC SNMP agent for two Pocket PC mobile devices accessing a wired network via an 802.11b wireless link. As a result of the implementation performed in this thesis, an SNMP manager can wirelessly communicate with a Pocket PC client. However, other results found that only some of the commercially available SNMP managers are able to access the mobile SNMP client and its management information base, due to incompatible implementations of the server and client software.


| **14. SUBJECT TERMS** Pocket PC, IEEE 802.11b, Wireless Network, SNMP, Windows CE 3.0, PDA, MIB, ActiveX, IPWorks! | | | **15. NUMBER OF PAGES** 108 |
|---|---|---|---|
| | | | **16. PRICE CODE** |
| **17. SECURITY CLASSIFICATION OF REPORT** Unclassified | **18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified | **19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified | **20. LIMITATION OF ABSTRACT** UL |

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release; distribution is unlimited**

SNMP OVER WI-FI WIRELESS NETWORK

Jiradett Kerdsri
Lieutenant, Royal Thai Air Force
BE(Civil), Air Force Academy (Thailand), 1998

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN COMPUTER SCIENCE**

from the

**NAVAL POSTGRADUATE SCHOOL**
**March 2003**

Author: Jiradett Kerdsri

Approved by: Ted Lewis
Thesis Advisor

Geoffrey Xie
First Reader

Gurminder Singh
Second Reader

Peter Denning, Chairman
Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

# ABSTRACT

Simple Network Management Protocol (SNMP) allows users of network equipment (i.e. Network Administrators) to remotely query the state of any device being tested for system load, utilization, and configuration.

Windows NT, Windows 2000 and Windows XP Professional are all equipped with SNMP service so that an SNMP manager can communicate with an SNMP agent running on a wireless 802.11b client. However the rest of Windows operating systems, including Windows CE and a Pocket PC, have to run third party proxy SNMP agents in order to be recognized by an SNMP management application.

This thesis describes an implementation of a Pocket PC SNMP agent for two Pocket PC mobile devices accessing a wired network via an 802.11b wireless link. As a result of the implementation performed in this thesis, an SNMP manager can wirelessly communicate with a Pocket PC client. However, other results found that only some of the commercially available SNMP managers are able to access the mobile SNMP client and its management information base, due to incompatible implementations of the server and client software.

THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF CONTENTS

## LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF ABBREVIATIONS, ACRONYMS AND SYMBOLS

| | |
|---|---|
| **ANSI** | American National Standards Institute |
| **ASN.1** | Abstract Syntax Notation One |
| **EVB** | Embedded Visual Basic |
| **FTP** | File Transfer Protocol |
| **ICMP** | Internet Central Message Protocol |
| **IEEE** | Institute of Electrical and Electronics Engineers |
| **IEFT** | Internet Engineering Task Force |
| **IP** | Internet Protocol |
| **ISO** | International Organization for Standardization |
| **LAN** | Local Area Network |
| **MIB** | Management Information Base |
| **NMS** | Network Management Station |
| **PDU** | Protocol Data Unit |
| **RFC** | Request for Comment |
| **RMON** | Remote Network Monitoring |
| **SMI** | Structure of Management Information |
| **SMP** | Simple Management Protocol |
| **SNMP** | Simple Network Management Protocol |
| **TCP** | Transmission Control Protocol |
| **UDP** | User Datagram Protocol |
| **WAN** | Wide Area Network |
| **Wi-Fi** | IEEE 802.11b Wireless Network |

THIS PAGE INTENTIONALLY LEFT BLANK

# I. PROBLEM

## A. OVERVIEW

The purpose of this thesis is to make a Pocket PC that is running on a wireless network visible on a Simple Network Management Protocol (SNMP) manager application. Because a Pocket PC or any Windows CE platform does not have SNMP service built in, the SNMP Proxy agent has to be implemented in software. The goal of this research is to implement SNMP services on handheld devices with 802.11 network capabilities and to test the performance on several commercially available devices.

One such device is the Personal Digital Assistant (PDA) that is increasingly becoming standard in many organizations. This handheld computing device, once used for simple tasks such as keeping addresses and appointments, can be used in the same way as a small notebook.[1] With the growth of new Wireless technology, more and more PDAs are equipped with a wireless interface and are used in the network environment. Therefore a PDA can now be considered as one of the networking devices. Since it fits in the wireless network due to the device's portability.

To manage the network environment, a SNMP is used. However the SNMP manage station can only discover and monitor the SNMP manageable device which is integrated with SNMP service; consequently all non-SNMP manageable devices are blind to the SNMP manager. Most wireless devices cannot be managed beyond the access point as well because they do not implement SNMP services. As the result a proxy agent for a PDA must be implemented to address this problem.

A proxy agent is the program used to support devices that do not have an SNMP implementation available. The proxy is an SNMP management agent that services requests from the management console, on behalf of one or a number of non-SNMP devices. The management station contacts the proxy agent and indicates the identity of the foreign device. The proxy agent then translates the protocol interactions it receives

[1] Tivoli Device Manager for Palm Computing Platform – Tivoli System Inc, and IBM company

from the management station into whatever interactions are supported by the foreign device.

## B. POCKET PC IN A WIRELESS ENVIRONMENT

More and more Pocket PC devices are connected to the wireless network since the widespread use of 802.11b standard; nevertheless some limitations exist for implementing a wireless application for Pocket PC. Although they share standard elements and mechanisms, wired and wireless networks have significant differences. In addition to the conventional wired network, wireless networks have the following unique properties:

### 1. Roaming Between Access Points

The wireless network environment supports a dynamic cell connection or roaming, which is the process of changing the network connection of a mobile device from one access point to another. This is a unique component within the wireless environment. Furthermore specifying an IP address for a roaming mobile device is difficult because SNMP relies on an UDP and a TCP/IP connection. Changing the IP address can corrupt communication between the Pocket PC agent and the network station.

### 2. Persistence of Mobile Units

Unlike desktop systems or other network components that operate continuously or are powered on/off daily, handheld terminals are turned on and off frequently throughout the day, making it difficult to monitor these devices. The intricacy of wireless communication is one of the big problems due to the persistence issue. The wireless communication for a Pocket PC takes a much longer time than the desktop PC to receive the wireless signal because of the hardware and software limitation. As a result, the SNMP management experiences difficulties since the SNMP manager can detect the negative fault due to the SNMP communication lags.

### 3. The SNMP Agent

Desktop or laptop systems have adequate amounts of memory and processor power to support an SNMP agent operating as a background task handling requests from

the Network Management station. However, on handheld terminals, typically running Windows CE, memory space and processor speed are highly limited resources, creating more effort in providing agents for these devices.

## C. SUMMARY

An SNMP Manager cannot see a Windows CE or Pocket PC device that is connected beyond the wireless access point because it lacks SNMP capability. To address this problem, an SNMP proxy agent has to be implemented in software, so the Network Management Station (NMS) can discover the device connected to the network. Additionally a developer faces many factors when implementing the wireless system on the limited device such as a Pocket PC.

Therefore in order to address the invisibility problem of a Pocket PC in wireless network management, this thesis is focused on implementing an SNMP proxy agent on Pocket PC, which can run on any Windows CE platform. The overview of the SNMP and the operation of a portable device are described in the following chapter. In addition, the architecture of the experiment and design plan are explained in the later chapter, including testing result.

THIS PAGE INTENTIONALLY LEFT BLANK

# II. THE OPERATION OF SNMP

## A. OVERVIEW

Since being developed in 1988, Simple Network Management Protocol has become the de facto standard for internetwork management. Because the protocol is a simple solution, requiring little code to implement, vendors can easily build SNMP agents for their products by various tools on the market. The protocol is extensible, allowing developers to easily add network management functions to their existing projects. In addition SNMP separates the management architecture from the architecture of the hardware devices, which broaden the base of multivendor support. Unlike other so-called standards, SNMP is not a mere paper specification, but an implementation that is widely available today.[2]

This chapter describes the basic elements on the SNMP framework as well as the transport operation for SNMP with TCP/IP. In addition this section also presented the overview and background of the network management.

Since SNMP is implemented on the User Datagram Protocol (UDP), it is a connectionless communication with no guarantees about the reliability of the transportation. The operation of SNMP communications requires Data Link Layer protocols, which is the layer 2 in the OSI model so it can use Ethernet or Token Ring to implement the communication channel from the management to the managed agent.

The two most important instances in the SNMP communication are manager station and managed agent:

A typical manager usually:

- Implemented as a Network Management Station (NMS)
- Implements full SNMP Protocol

[2] SNMP,http://www2.rad.com/networks/1995/snmp/snmp.htm [02]

- Be able to
    - Query agents
    - Receive Get responses from agents
    - Establish Set variables in agents
    - Acknowledge asynchronous events from agents

A typical agent usually:

- Implements full SNMP protocol.
- Stores and retrieves management data as defined by the Management Information Base
- Can asynchronously signal an event to the manager
- Can be a proxy for some non-SNMP manageable network node

A MIB is a collection of definitions, which define the properties of the managed object within the device to be managed.

Criteria and Philosophy for standardized MIB are

- Objects have to be uniquely named
- Objects have to be essential
- Abstract structure of the MIB need to be universal
- Standard MIB maintain only a small number of objects
- Private extensions allowance.
- Objects must be general and not overly device dependent
- Objects can not be easily derivable from their objects
- If agent is to be SNMP manageable then it is mandatory to implement the Internet MIB (MIB II in RFC 1157)[3]

[3] SNMP, http://www2.rad.com/networks/1995/snmp/snmp.htm [02]

## B. NETWORK MANAGEMENT ARCHITECTURE

The big picture of SNMP network management architecture is shown in Figure 1 and Figure 2. Figure 1 describes the hardware architecture in the SNMP network management consisting of Network Management System (NMS), Wired and Wireless clients. Figure 2 explains the SNMP communication software in the system.

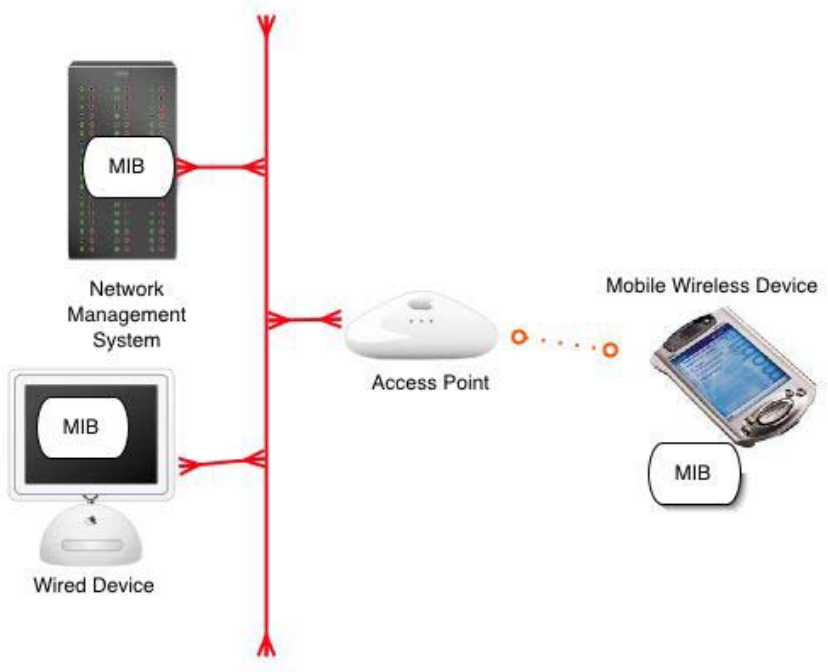



**Figure 1 – SNMP Hardware Architecture.**

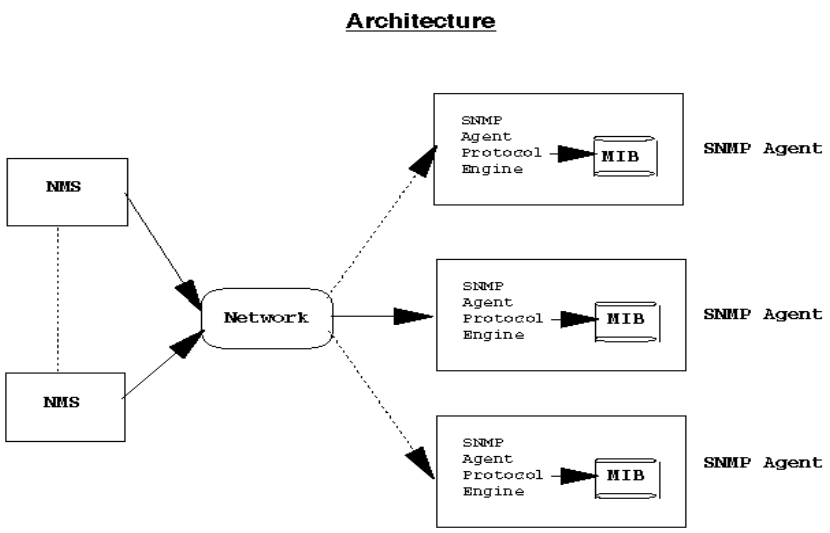



**Figure 2 – SNMP Software Architecture From Ref.[02].**

The models of network management that is used for the popular TCP/IP network management are usually designed with the following key elements in Figure 3 – Transport Mechanisms.

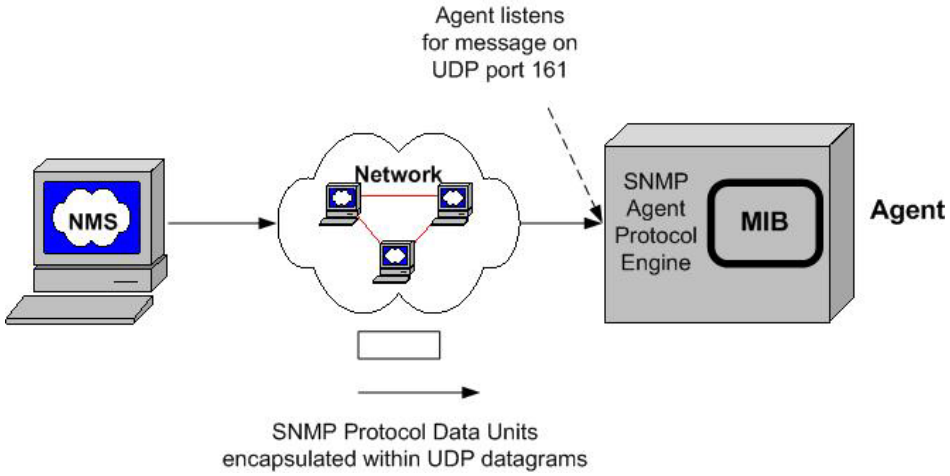



**Figure 3 – Transport Mechanisms.**

There are four major elements in SNMP network management:

**1. Network Management Station**

Network Management Station (NMS), typically a stand-alone device, provides the interface for a human network manager to interact with the management system. The management station has the capability to configure, monitor, analyze and control the various components that comprise the network. Some prominent vendors offer network management platforms which implement the role of the network management manager, such as Hewlett-Packard OpenViewTM , IBM NetView, 3 Com Network Supervisor.

Both commercial and non-commercial SNMP Managers are available. Some of the SNMP managers are proprietary and available with the purchased hardware. For this experiment three SNMP Manager products were used to test the SNMP proxy agent; Loriot, SNMPView, and 3COM Network Supervisor. The Loriot and 3COM Network Supervisor are the full-package standard SNMP software for the enterprise while SNMPView is an small open source software that is configurable.

***a. Loriot***

Loriot is a Free, GUI SNMP V1, V2c and V3 node Manager running over MS-Windows 95/98/NT4/2000/XP small platform. Loriot is designed for managing the SNMP for a small to large sized network. Loriot manages hosts, routers, links and all SNMP equipment of LAN/MAN/WAN networks, and it is able to manage them through a customizable graphic representation of the organization. Loriot supports many classic management processes, such as customizable events polling, MIBs compiler, script query language, network discovery, automatic report generation, etc. Events are customizable and associated with actions through an advanced filter. Supported actions associated with filters are Winrun, Dosrun, Wave, Syslog, reroute Trap, SMTP. Loriot supports plug ins, like device GUIs, Cisco ISDN call management, and Cisco configuration management. One SDK running on Visual Studio 6.00 with Wizard modules and sources is available to create the plug in. Loriot does not use WinSnmp.dll. Instead, it uses its own SNMP API based on a new SNMP C++ class called DSNMP. This is simple to use but very powerful. Loriot includes a HTTP daemon module to support scalability through a remote HTPP/JAVA console with the capacity to reroute the local events like SNMP

Trap through an advanced filter. In this way distributed events are displayed on the master Loriot console.[4]

***b. SNMPView***

The SNMP View from a non-profit company in Holland (http://www.snmpview.de/index.php) is open source SNMP manager software that is simple to use and configure. This free software frequently queries network devices and shows values that are readable over SNMP in a table. Thresholds can be watched and colorized at violation. User can transform cryptic numbers in human readable text by using a translation table. The software can write a log file with all values. All parameters are freely configurable as a text file.

***c. The 3 COM Network Supervisor***

The 3Com® Network Supervisor is a powerful easy-to-use management application that graphically discovers, maps, and displays network links and IP devices, including 3Com NBX® telephones and some popular third-party products. It maps devices and connections, so it can easily monitor stress levels, set thresholds and alerts, view network events, generate reports in user-defined formats and launch device configuration tools. Moreover when the network changes, the user can prompt the 3Com Network Supervisor to regenerate the appropriate part of the map to ensure that the user has current information.[5]

**2. Management Agent**

The management agent is the second active element in the management architecture; it is the workhorse of the SNMP communication. The agent is a software program in the network device that responds to requests for information or actions issued by the management station. The agent may also send the station unsolicited information, known as “Trap”. All devices in a network must have a management agent. Typically, an agent may be embedded or "native" to the device, or alternatively be a "proxy" agent for other protocols.

---

[4] http://www.llecointe.com/index.html , accessed on 5 March 2003

[5] http://www.3com.com/product/en_US/productlist.jsp? , access on 5 March 2003

A proxy agent is the program that supports devices without available SNMP implementation (most mobile devices on the market do not have SNMP implementation) .The proxy is an SNMP management that services requests from the management console, on behalf of one or a number of non-SNMP devices.

### 3. Management Information Base (MIB)

The third part of the architecture is the information exchanged between the manager and the agent; this is called the Management Information Base or MIB. This information is a collection of objects or data values with each representing one aspect of the managed device. For example, the location of the device and the number of erred seconds in the last hour would be two different data values in the MIB. The structure and content of the MIB are standardized across systems of a particular class, such as a bridge MIB or DS-3 MIB. After a MIB is published as a standard, various vendors can build the same kind of equipment that complies with the MIB being assured that they can be managed in a TCP/IP network.

The MIB structure is standardized in SNMP as a hierarchical tree. Additions to the tree can be easily accomplished, while traversing a tree to obtain specific information can be done very quickly. These are important features because they encourage the use of the MIB in the network management model and the creation of enterprise MIBs for vendors looking to support SNMP with their own products.

The structure of Management Information (SMI) which is given in RFC 1155, is based on the OSI SMI given in Draft proposal 2684. The latest Internet MIB ,given in RFC 1213, is called “MIB II”. The SMI states that each managed object must have the following elements: a name, a syntax and an encoding.

- The name, an object identifier (OID), uniquely identifies the object.
- The syntax defines the data types, such as integer or string of octets. The syntax used for SNMP is the Abstract Syntax Notation One (ASN.1).

- The encoding describes how the information is associated with the managed objects. The encoding used for SNMP is the Basic Encoding Rules (BER).[6]

The popularity of SNMP has resulted in the development of standards for storing data critical to network operation, the Management Information Base (MIB). MIB-II, the latest generation of network management MIBs, stores data on TCP/IP traffic, routing, configuration and errors. MIB-II has improved support for multi-protocol devices and allows the network management system to control SNMP operation.

A Pocket PC 2002 or Windows CE 3.0 includes SNMPv2c extensible agent support, Management Information Base (MIB) II (TCP/IP stack), Host MIB, and a sample MIB. Because SNMP is compatible with Windows NT source code, MIBs can be ported from the Windows NT familty.

**4. Network Management Protocol**

The last element of the model, the network management protocol, links the station and the agent by specifying the rules for communication. The protocol used for the management of TCP/IP networks is the SNMP, which uses three simple commands to communicate:

- Get enables the station to retrieve data from the agent
- Set enables the station to set a value at the agent
- Trap that enables the agent to notify the station of an important event.

Other protocols are available, such as Internet Control Message Protocol (ICMP) and Simple Gateway Monitoring Protocol (SMGP), but they have limited functionality and only support a generic MIB. As a result, these two protocols are not widely used.

[6] SNMP, http://www2.rad.com/networks/1995/snmp/snmp.htm [02]

## C. NETWORK MANAGEMENT

In the rapid pace of growing technology, networks tend to be large and complex, filled with many types of equipment from different vendors. Managing such a network is increasingly more difficult, involving multiple management tools and protocols to support different proprietary devices on the network. Therefore, the network administrator needs a network management tool that can monitor the network's availability, utility and performance with the most industry-acceptable standards as possible to reduce the conflict of the network management system. Network management is the collection of tasks performed to maximize availability, performance, security and control of a network and its resources.

The purpose of network monitoring is to gather information about the status and behavior of network elements. Information to be gathered includes static information, related to the configuration; dynamic information, related to events in the network; and statistical information, summarized from dynamic information.

Typically, each managed device in the network includes an agent module responsible for collecting local management information and transmitting it to one or more management stations. Each management station includes network management application software plus software to communicate with agents. Information may be collected actively, by means of polling by the management station, or passively, by means of event reporting by the agents.

The International Organization for Standardization (ISO) Network Management Forum has divided network management into five functional areas:

### 1. Fault Management

The objective of fault monitoring is to identify faults as quickly as possible after they occur and to identify the cause of the fault so that remedial action may be taken. Comprehensive fault management is the most important task in network management. Fault management tools can help increase the reliability of the network by quickly identifying the fault, isolating the cause of the fault, and then, if possible, correcting the fault.

**2. Configuration Management**

Configuration management deals with the initialization, modification, and shutdown of a network. Networks are continually adjusted when devices are added, removed, reconfigured, or updated. The process of configuration management involves identifying the network components and their connections, collecting each device's configuration information, and defining the relationship between network components. In order to perform these tasks, the network manager needs topological information about the network, device configuration information, and control of the network component.

**3. Performance Management**

Performance management involves measuring the performance of a network and its resources in terms of utilization, throughput, error rates, and response times. With performance management information, a network manager can reduce or prevent network overcrowding and inaccessibility. This helps provide a more consistent level of service to users on the network, without overtaxing the capacity of devices and links.

**4. Accounting Management**

In the area of accounting management, network monitoring is concerned with gathering usage information to the level of detail required for proper accounting. This type of information helps a network manager allocate the right kind of resources to users, as well as plan for network growth. This type of management also involves monitoring access privileges and usage quotas of the users.

**5. Security Management**

Security management deals with ensuring overall security of the network, including protecting sensitive information through the control of access points to that information. Sensitive information is any data that an organization wants to secure, so protecting this sensitive data from unauthorized access is a common requirement. Security concerns can be assuaged with a well-designed and implemented security management system.[7]

[7] SNMP,SNMPv2,SNMPv3, and RMON1 and 2 – William Stalling, Addison-Wesley, 1999 [04]

All these types of functional areas are instrumental to network management. However, the research in this thesis focuses on only two types of the functional areas : Fault Management and Configuration Management which are the basic utility of most SNMP application.

## D. NETWORK MANAGEMENT STANDARDS

The Internet Architecture Board (IAB), which is responsible for networking technology and protocols for the TCP/IP internetworking community, has created a Standard Network Management Protocol (SNMP). This protocol is documented as Request For Comments (RFCs) and is widely published.

The IAB recommends SNMP for use as a common network management protocol with TCP/IP-based networks. Networking with TCP/IP is the most popular type of network and internetwork. As described by RFC 1157, SNMP was initially specified in the late 1980s with three later enhancements that solidified the role of SNMP as the indispensable network management tool

An enhanced version of SNMP, called SNMPv2 was released in 1993 and revised in 1995. The latest standard called SNMP3, issued in 1998, defines an overall framework for present and future versions by adding security features to SNMP

## E. SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP)

Simple Network Management Protocol (SNMP) is a distributed-management protocol. A system can operate exclusively as either an NMS or an agent, or the protocol can perform the functions of both. When a system operates as both an NMS and an agent, another NMS might require that the system query manage devices and provide a summary of the information learned, or that it report locally stored management information.[8]

Since SNMP is the only protocol for the management of TCP/IP networks, three other specifications are necessary to form the foundation of the management system. Together, these are

[8] SNMP, http://www.cisco.com/unvercd/cc/td/doc/cisintwk/ito_doc/snmp.htm [6]

- RFC 1157: A Simple Network Management Protocol that defines the protocol and architecture used to manage the MIB and its contents;
- RFC 1155: Structure and Identification of Management Information for TCP/IP-based networks that describe how the MIB is defined;
- RFC 1213: Management Information Base for Network Management of TCP/IP-based Internets: MIB-II that describe the contents of the MIB.

This application layer protocol, SNMP, is part of the Transmission Control Protocol/ Internet Protocol (TCP/IP) protocols' suite. It is intended to operate over the User Datagram Protocol (UDP). The management station communicates management information using SNMP, which is implemented on top of the UDP and IP protocols, and a data link protocol, such as Ethernet, Token Ring, or X.25. Likewise, the agent must also implement SNMP, UDP, IP protocols.

Several key features of SNMP important to understand are transaction-based and datagram-oriented protocol. The protocol dictates that an action gets a response that ends the transaction. The response is a datagram or packet of bytes of prescribed size and format. Action by the management station, such as a GET command, will result in a datagram response from the agent. If the transmission fails to reach its destination address, the management station will retransmit again.

A strong focus for SNMP is Configuration and Fault Management. Since protocol has fewer features for Security management, most SNMP management systems concentrate on alarm and status reporting and use other methods for security. In addition, Trap commands sent by the agent to the management station are less reliable than Get responses because the UDP protocol is used. Nevertheless UDP does not guarantee delivery of the message. An alarm notification or Trap from a faulty device may be lost because the datagram failed to reach the management station. [9] However SNMP, actually transported independently, can be implemented on other transports as well such as TCP, direct mapping on to Ethernet MAC level, X25 protocol or ATM cell encapsulation.

[9] Network Management , http://www.airlinx.com/index.cfm/id/1-11.htm [3]

### 1. SNMP Message Construct

Each SNMP package has the filed that creates a message construct. The format includes

Version Number

Community String

One or more SNMP PDUs (data which is a sequence of PDUs associated with the request).

The detail of a SNMP message as in Figure 4 is implemented in a UDP package communicating on Ethernet. Each SNMP data consists of fields that depend on the type of command and version of SNMP. See Figure 5 through Figure 8 that describe the format for SNMP message.

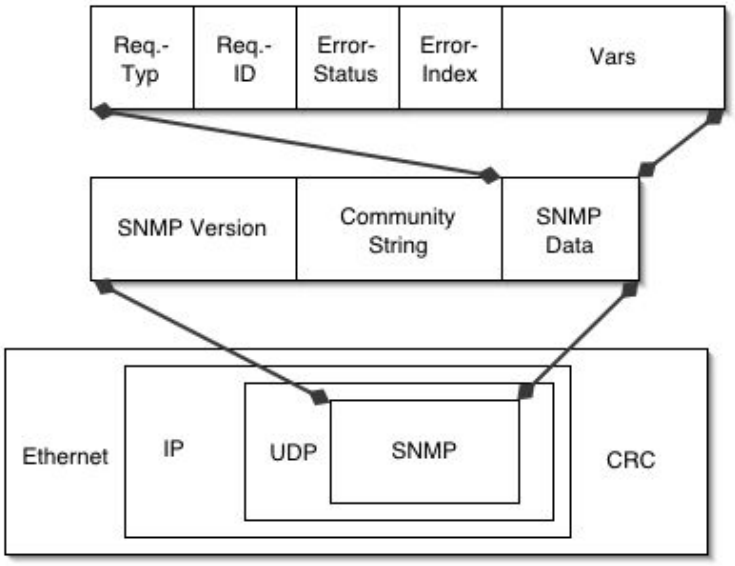



**Figure 4 – SNMP message From Ref.[03].**

Each SNMP PDU, except Trap, has the following format:

PDU type – Specifies the type of PDU transmitted.

Request ID – Associates SNMP requests with response.

Error Status - Indicates error and error types.

Error Index - Associates an error with a particular object instance.

List of OIDs and their values – Associates with each variable binding a particular object instance with its current value (with the exception of Get and GetNext requests, for which the value is ignored)

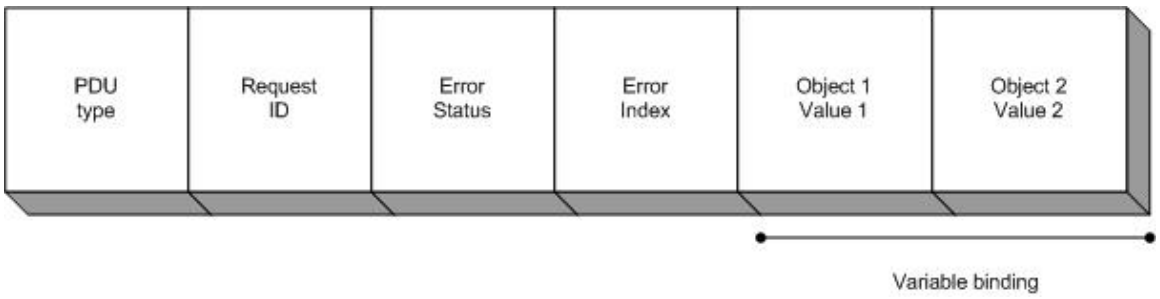



**Figure 5 – SNMPv1 Get, GetNext, Response, and Set PDUs Contain the Same Fields From Ref.[03].**

Trap PDUs have the following format

Enterprise – The types of object causing the Trap

Agent address – IP address of agent which sent the Trap

Generic Trap ID – The common standard Trap commands

Specific Trap ID – Proprietary or enterprise Trap

Time Stamp – Trap occurs in time ticks

List of OIDs and values – OIDs that may be relevant to sent to the NMS

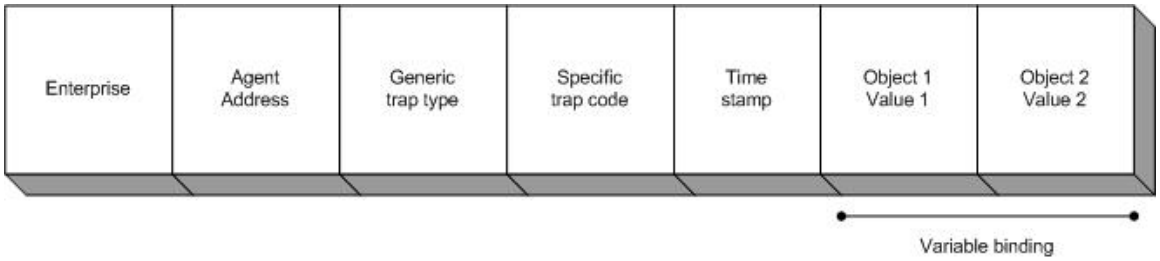



**Figure 6 – SNMPv1 Trap PDU From Ref.[03].**

For SNMPv2, however, Trap PDU has the same field as the other commands, but it has an additional GetBulk PDU with the a field.

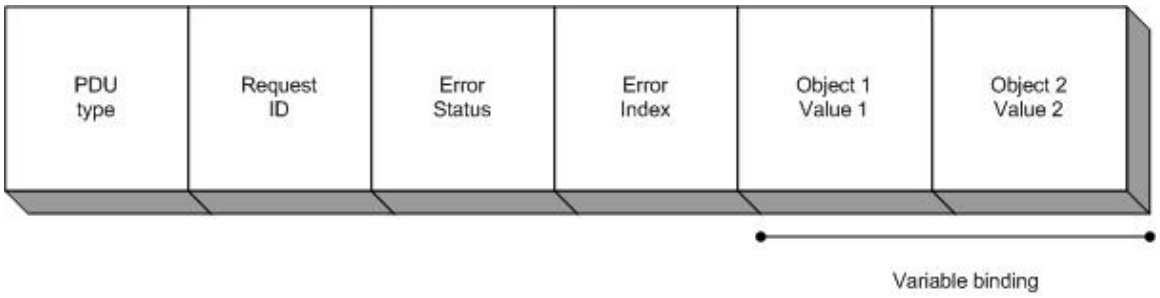



**Figure 7 – SNMPv2 Get, GetNext, Inform, Response, Set, and Trap PDUs Contain the same fields From Ref.[03].**

The GetBulk PDU format consists of these following elements:

PDU type – Identifies the PDU as a GetBulk operation.

Request ID – Associates SNMP requests with response.

Non repeaters – Specifies the number of object instances in the variable bindings field that should be retrieved no more than once from the beginning of the request. This field is used when some of the instances are scalar objects with only one variable.

Max repetitions – Defines the maximum number of times that other variables beyond those specified by the Non repeaters field should be retrieved.

Variable bindings – Serves as the data field of the SNMPv2 PDU. Each variable binding associates a particular object instance with its current value (with the exception of Get and GetNext requests, for which the value is ignored).

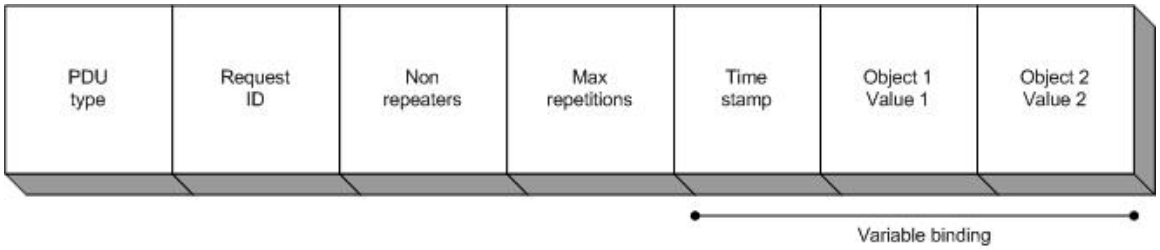



**Figure 8 – The SNMPv2 GetBulk PDU From Ref.[03].**

## 2. SNMP Security

Since SNMP lacks any authentication capabilities, this results in its vulnerability to a variety of security threats. These include masqueraded occurrences, modification of information, message sequence and timing modifications, and disclosure.

Masqueraded occurrences consist of an unauthorized entity attempting to perform management operations by assuming the identity of an authorized management entity.

Modification of information involves an unauthorized entity attempting to alter a message generated by an authorized entity so that the message results in unauthorized accounting management or configuration management operations.

Message sequence and timing modifications occur when an unauthorized entity reorders, delays, or copies and later replays a message generated by an authorized entity.

Disclosure results when an unauthorized entity extracts values stored in managed object, or learns of events by monitoring exchanges between managers and agents.

Therefore many vendors do not implement the “SET” operation because SNMP does not have enough authentication security, thereby reducing SNMP to a monitoring facility. However some features added to SNMPv3 including security functionality and access control even though does not widely use yet.[10]

[10] SNMP, http://www.cisco.com/unvercd/cc/td/doc/cisintwk/ito_doc/snmp.htm [6]

THIS PAGE INTENTIONALLY LEFT BLANK

# III. SOFTWARE DEVELOPMENT TOOLS FOR PORTABLE DEVICES

## A. OVERVIEW

There are many kinds of Personal Digital Assistant (PDA) devices on the market but only two mostly dominate the market, Palm PDA and its competitor, Pocket PC.

Palm handhelds have made a big impact in the world of portable organizer and have become extremely popular over the past few years because they are easy to use and easy to carry. All Palm OS® devices usually come with four major standard software applications, the address book, date book, memo pad, and To-Do list. These applications provide enough functionality for Palm to organize people. Palm is not intended to replace a desktop or laptop, with their full environment, but is designed to be a satellite-computing device supporting people while they are away from their desk.

The Pocket PCs are handheld computers that go far beyond the capabilities of traditional PDAs. Pocket PC is superior to the Palm in performance because the Pocket PC usually has the faster processor, more memory and higher screen resolution than Palm devices. Furthermore the Pocket PC includes the Pocket versions of familiar Microsoft applications such as Word, Outlook and Excel so a Pocket PC user can synchronize their work between Microsoft applications without worrying about importing to another format.

In addition, the rest of the portable embedded operating systems have also made a big impact on the small devices, such as Symbian, an advanced open standard operating system for data enable mobile phones or embedded Linux, the open source operating system that can integrated with many kinds of devices, such as networking devices, smart phone or PDA.

## B. THE POCKET PC

The Pocket PC operating system, created by Microsoft, is an evolved version of Microsoft's Windows CE operating system that has been around since 1996. Even though a wide range of mobile devices could technically be considered Pocket PCs, the phrase "Pocket PC" refers to a very specific class of devices that adheres to the hardware

standards set forth by Microsoft. Microsoft defines these standards because it makes the operating system for the devices. The Pocket PC operating system should be familiar to anyone who has experience with the Windows operating system family. In fact, the idea behind Pocket PC is to translate the success of Windows to the handheld computer market. The result of this translation is the Pocket PC 2002, which is now the standard operating system for Pocket PC PDA (as of writing February 2003) 11

The Pocket PC Operating System was born under the name Windows CE in 1994 when Microsoft began developing a PDA known as WinPad. The WinPad initially was conceived to compete with Apple Newton, a PDA that eventually failed to be a commercial success due to numerous problems. This led Microsoft to realize that affordable technology was not quite there to make WinPad a success. As a result, Microsoft cancelled the WinPad project and abandoned the thought of a mobile operating system, at the time.

Microsoft's latest attempt at a compact operating system, under the code name Pegasus, was publicly unveiled as Windows CE 1.0 at Fall Comdex 1996. By trying to use a Graphic User Interface (GUI) from the big success of Windows 95 on the Windows CE 1.0. Pegasus was not considered a real PDA because it did not have the handwriting-recognition software while most of the other PDAs in the market used some form of handwriting- recognition.

Soon after Microsoft released Windows CE 2.0, Microsoft released a much improved version, Windows CE 2.1. However both versions of the Operating System still suffered because the users did not want to use the Windows 95 interface on anything but their desktop or notebook computers.

The third attempt at getting Windows CE to work correctly and to be accepted by users resulted in Windows CE 3.0 or "Pocket PC", which was a big hit on the PDA road of Microsoft. By changing the name and cleaning the interface, Microsoft gained the recognizable brand with no negative history to drag it down. The Pocket PC has two versions, Pocket PC 2000 and the latest Pocket PC 2002.

11 Using Pocket PC 2002 – Michael Morrison, Que Publishing, June 2002 [01]

Pocket PC 2002 represents the incremental improvement in the world of PDA with all new clean features that fit in to the mobile devices. Microsoft has gained market share because of the following improvements.

**1. Pocket PC Structure**

The Pocket PC (or Windows CE) is an object-based operating system. Every resource, such as a process, thread, or file, that a program creates or accesses appears as a translucent object to the application. The Object Manager performs the crucial task of relating the data structures in the operating system memory to a translucent object handle or identifier. In this way, a client application can only access the resource through a set of controlled methods that require the object handle and validate the input. Additionally, the client application cannot directly modify the operating system data structure. The ultimate result of all these object-based protections is to impart a greater reliability to the client application and to the operating system itself.

When a user initiates the execution of a client application, the Process Manager comes into play. The component of the operating system creates an initial thread for the application, called the "primary thread", and establishes a number of important data structures for the application, such as the initial memory heap. In fact, the process Manager creates these resources through interactions with other components: the Memory Manager and the Kernel.

***a. Memory Manager***

The allocation, deallocation and tracking of all available physical memory are the functions that the Memory Manager performs. When the Process Manager or a client application requests memory, the memory Manager finds the available memory, marks the memory as being allocated, and assigns the memory to the application. Upon release of the memory, this component simply reverses the processes, releasing the memory for use by other client application.

For the Pocket PC, the file space comes from the available memory inside the Pocket PC rather than an external device. Therefore, the File Manager interacts with

the Memory Manager to allocate the memory and then creates the data structures to manage the file, such as the pointer to the current location being accessed in the file.

***b. Kernel***

The most important member of the Windows CE operating system is the Kernel. The primary job of the Kernel is to manage and schedule the set of existing executing threads. When the Process Manager requests the creation of an application's primary thread, the Kernel actually establishes the thread and creates the necessary data structures. Of course, the allocation of the data structures occurs by interaction with the Memory Manager. Two sets of data structures are necessary for each thread. One data structure enables the Kernel to maintain the state of the executing thread during CPU sharing by maintaining values such as the program counter of the thread code. The other primary data structure is the stack that the thread uses to allocate and manage local variables. [12]

**2. Pocket PC and Wireless**

In the past, Pocket PC could only connect to the network and internet wirelessly through a mobile phone, which has a built in modem, via infrared. However the new technology of the wireless communication has changed in the past few years because of IEEE 802.11 standard.

The popularity of the wireless networking especially the IEEE 802.11b protocol has brought new excitement to Pocket PC handheld devices because Pocket PC can now take the full advantage of wireless communication. Because Pocket PCs are in fact mobile computers, Pocket PCs can connect to the network in many ways by using many kinds of network interfaces. With the compact flash wireless card for the Pocket PC with compact flash slot or wireless PCMCIA card for the Pocket PC with PCMCIA slot, Pocket PCs can connect to wireless as easily as desktop or notebook computers. Moreover most of the new versions of Pocket PCs, such as Toshiba E740 or IPAQ 5450 are integrated with 802.11b wireless.Instead of taking notebook to the Wi-Fi hotspot to check email or use the Internet, Pocket PC can do everything that a notebook can do plus

[12] Pocket PC Developer's Guide, Bruce E. Krell Ph.D., McGraw-Hill 2002 [08]

it is easier to carry and to turn power on and off. In the very near future, people will use Pocket PC as a convenience device to send wireless email or instant messaging.

### 3. Pocket PC Programming

The Pocket PC programming has a lot of criteria to be considered. The types of CPU and Operating Systems are the most significant parts to analyze. In the past few years, Pocket PCs from different venders came with different kinds of CPUs such as NEC MIPS, which is usually used in Casio Pocket PC or Hitachi SH3 for HP Jornada. Consequently the developers must implement an application that supports all the CPU platforms. Moreover the users must download and install the right application for their CPU types. However nowadays most Pocket PCs come with Intel ARM Processor or Intel Xscale Processor which is effortless for the developer because most of the applications written for ARM CPU are compatible with Xscale CPU.

The type of Operating System is not an immense issue for Pocket PC because most of applications designed for Pocket PC 2000 are compatible with Pocket PC 2002 although the applications for Windows CE are not compatible.

Numerous approaches exist to write the application on Pocket PC or Windows CE operating system, but the most popular tool is the Embedded Visual Tool from Microsoft. Java, a competitor from Sun Microsystems, is also popular among the Java users and Web application developers. The latest tool that Microsoft hopes to compete with Java, Microsoft .NET framework, also is a good tool combining all the good things aspects in Embedded Visual Tools and Java.

#### *a. Embedded Visual Tools*

An Embedded Visual Tool consists of Embedded Visual Basic (EVB) and Embedded Visual C++ (EVC); therefore, it is straightforward for the developer who is familiar with Visual Studio application on the desktop PC. Embedded Visual Studio version 3.0 consists of Embedded Visual Basic 3.0 and Embedded Visual C++ 3.0; furthermore, the latest version of Embedded Visual C++, released since the middle of 2002, is version 4.0. Embedded Visual C++ 4.0 is backward compatible with Embedded Visual C++ 3.0 and compatible with Windows CE.Net, the newest Operating System for

portable device from Microsoft. On the other hand, the Mobile Operating System supported by Embedded Visual Studio 3.0 is Pocket PC 2000. In order to implement application on Pocket PC 2002, a Pocket PC 2002 SDK is required.

However Microsoft released the new version of Embedded Visual Tool that integrated Pocket PC 2002 SDK and Smartphone 2002 SDK on December 2002. Hence Embedded Visual Tool is the perfect environment software to implement an application on Pocket PC 2002 platform. (All the tools available to downloaded from www.pocketpc.com)

***b. Personal Java***

Java, the other alternative for programming on Pocket PC ,which was developed by Sun Microsystems, is a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high-performance, multithreaded, dynamic, general-purpose programming language. In order to execute Java application, Java Runtime Environment (JRE) is required; therefore, every machine on any operating system that has JRE could run Java Application including Pocket PC and Palm. The "Write once run anywhere" is the advantage of Java over the other languages. There are two Java platforms in the portable device deployed environment: Java 2 Micro Edition (J2ME) and Personal Java.

Java 2 Platform, Micro Edition is based on the mobile platforms, such as pagers, cell phones, and other small-scale devices. Basic J2ME is a minimalist set of Java APIs, while profiles add additional API capabilities. The basic J2ME runtime is the Connected Device Configuration (CDC), which contains the virtual machine and necessary pieces to run the system.

In reality Sun Microsystems announced Personal Java prior to J2ME. Personal Java defines a different subset of the core Java APIs for use within reduced memory footprint devices, like tabletop screen phones, high-end PDAs, and set-top boxes. This is used in devices typically larger than those used for J2ME but not quite up to par as current desktop standards go.

Personal Java includes most of the necessary Abstract Windowing Toolkit (AWT) components set with support for applets as well as network connectivity. There is even optional support for Java Database Connectivity (JDBC), Remote Method Invocation (RMI), and compression (java.util.zip) to meet distributed programming needs. Although Personal Java does not support the Swing component, all the provided features are good enough for Pocket PC application development.

Sun Microsystems has discontinued supporting Personal Java since 1999; however, Insignia, a lead company in Java-enabled mobile devices area (http://www.insignia.com/content/products/pda.shtml) continues to develop Personal Java and to release a product called Jeode, which helps the developer continue to employ Java programming on embedded device.

The Jeode application environment is designed to run with a wide range of operating systems and target processors. It currently supports operating systems, such as Pocket PC 2002, Pocket PC 2000, Windows CE, Windows NT4, VxWorks, Linux, ITRON, Nucleus, BSDI Unix and pSOS; and processors, such as MIPS, ARM, Intel x86, PowerPC and Hitachi SH-3 and SH-4. As a result, Jeode is an excellent alternative to deploy the application written in Java to the Windows CE or Pocket PC environment because of the compact and versatile Java open source Platform.

### *c. .NET Compact Framework*

The latest option for Pocket PC application development is the Visual Studio .NET and .NET Compact Framework platform. By using the same idea of JRE for a multi-platform operating system, Microsoft released the Microsoft .NET framework in mid of 2002.

Microsoft has been successful in the desktop operating system and application market for many years but has yet to achieve similar success in the areas of large-scale enterprise and Internet systems, highly distributed systems, and devices. As the result, the .NET initiative is Microsoft's attempt to make greater inroads into this market.

Microsoft .NET is ambitious inventiveness with the goal of providing an integrated platform for the creation, distribution and execution of applications.

Comparisons of Java and .NET are common. Although it is undeniable that the Java platform provides the foundation for the primary competition that .NET will face, the scope of the .NET scheme is much greater because of the limitation of Java in Windows base programming. While both Java and .NET have the same slogan "write once, run anywhere," .NET provides a faster and more compatible application for Windows Operating System.[13]

Microsoft released .NET framework for Windows Operating System in 2002 and .NET compact framework for Windows CE and Pocket PC in 2003. The .NET framework for other operating systems has not been released yet, even though there are a lot of companies volunteering to develop .NET framework for other operating system. The .NET Compact Framework is the smart device development platform, and it is the subset of the desktop .NET framework, thus developers can easily reuse existing programming skills and existing code throughout the device, desktop, and server environment. Microsoft released the add on for Visual Studio .NET called Smart Device Extensions (SDE), which let the developers implement the application for Pocket PC and Windows CE on Visual Studio .NET. Moreover Microsoft has a plan to launch Visual Studio .NET 2003 which will be integrated .NET Compact Framework tools in mid 2003.

**4. Pocket PC and SNMP**

There is little information available about the SNMP development on both Pocket PC and Palm platforms, but Pocket PC seems more compatible with SNMP. Because Palm OS has been targeting the enterprise environment for many years, most of the SNMP implementation has been focused on Palm OS for the last few years. Even the big companies in the SNMP area such as Tivoli IBM has released "Tivoi Device Manager for Palm Computing Platform" to manage Palm devices in the enterprise, which contain both an NMS and an extensible agent.

Most SNMP developers implement SNMP NMS for Pocket PC. For example Netaphor, PDAlerts, SNMP manager applications run on an ARM processor. It can discover the network managed devices' condition and notify the manager of the situation. So the manager can monitor the status of the network from anywhere wirelessly.

[13] C# for Java Developers, Allen Jones, Microsoft press 2003 [07]

Windows CE does support the SNMP-agent application and the SNMP-agent application residing in the Windows CE-based device. The SNMP-agent support that is provided by Windows CE consists of a master agent or SNMP device, and one or more extension agents. The master agent receivers requests from the manager, decodes them, and delivers them to an extension agent by invoking the appropriate extension-agent dynamic-link library (DLL). To retrieve the information that is requested, the extension agent implements the appropriate MIB that contains the definitions of a set of managed objects supported by the extension agent.

Windows CE supports SNMP extension-agent and utility application programming interfaces (APIs). The SNMP extension-agent API functions define the interface between the SNMP service and SNMP extension-agent DLLs. The SNMP utility API functions facilitate memory management and simplify manipulation of SNMP data structures. Windows CE 3.0 supports two versions of SNMP: SNMPv1 and SNMPv2C. The first version of the protocol, SNMPv1, is defined by RFC 1157, RFC 115, and RFC 1212 whereas, SNMPv2C is the version of the protocol that is known as community string-based SNMPv2. It is defined by RFC 1901, RFC 1905, and RFC 1906. SNMP can be implemented in a Windows CE-based platform only if the Point-to-Point Protocol (PPP) module is included in the same platform.[14] However the difficulty of implementing such an SNMP agent on Windows CE 3.0 is the developer must understand the Windows CE operating system and C++ in depth. Moreover for this research, an agent that can run on any Pocket PC platform needs to be implemented. Hence the agent that is written on the third party standard framework is more practical for this research.

### 5. Pocket PC and ActiveX Control

The software framework for this research is implemented using the IP*Works! ActiveX edition from nsoftware. An ActiveX Control is optimized for the Internet application with out dependencies on external libraries. In fact an ActiveX Control is just another term for “OLE Object” or, more specifically, “Component Object Model (COM) Object.” In other words, a control, at the very least, is some COM object that supports the

[14] Writing your own SNMP Management Information Base Microsoft Windows CE 3.0, Microsoft Corporation, July 2000 [09]

IUnkown interface and is also self-registering. It usually supports many more interfaces in order to offer functionality, but all additional interfaces can be viewed as optional and, as such, a container should not rely on any additional interfaces being supported. By not specifying additional interfaces that a control must support, a control can efficiently target a particular area of functionality without having to support particular interfaces to qualify as a control. As always with OLE, whether in a control or a container, it should never be assumed that an interface is available, and standard return-checking conventions should always be followed. It is important for a control or container to degrade gracefully and offer alternative if a required interface is not available. For a control to operate well in a variety of containers, the control must be able to assume some minimum level or functionality in all containers. An ActiveX control object can register on a Pocket PC by adding to the Pocket PC 2002 control manager in Embedded Visual Tool. However Pocket PC 2002 emulator on Embedded Visual Studio has a conflict with ActiveX component from IP*Works!, therefore the experiment has to be tested on a Pocket PC device.[15]

## C. IMPLEMENTING SNMP AGENT

At least three approaches are available to implement SNMP agent for Microsoft Pocket PC. For C++ developers, Windows CE 3.0 already supports the SNMP agent-application; therefore, the developers can follow the white paper from Microsoft "Writing your own SNMP MIB for Microsoft Windows CE 3.0." To implement MIB and the SNMP agent on Pocket PC, however this subject requires high level of C++ programming skill.

The Java Dynamic Management Kit (JDMK), a product from Sun Microsystems, can assist Java developers to create an agent application that is both an SNMP agent and a JMX agent. This is the easiest way to implement the agent on a Windows CE based device.

Microsoft .NET compact framework is the best choice to implement the network application for Pocket PC because of its compact, speed and compatibility with the

15 http://msdn.microsoft.com/library/default.asp?url=/workshop/components/activex/intro.asp , access on March 2003

Windows CE structure. In addition .NET compact framework is integrated with most of the packages that developers needed. Additionally Visual Studio .NET is an excellent tool for developers to implement the application for multiplatform operating system because it is easy to use and consist of a large number of features. However as of implementing such an application, the Microsoft Visual Studio .NET 2003 is a beta version and still has some problem with communication between connected devices.

Hence Microsoft Visual Tool seems to be the most excellent software to implementing SNMP agent on Pocket PC 2002 so far. The ActiveX control can be registered to the Pocket PC device by adding a controller to the control manager for Pocket PC 2002 device. The agent is written with Embedded Visual Basic (EVB) and uses the ActiveX component from IP*Works!

THIS PAGE INTENTIONALLY LEFT BLANK

# IV. EXPERIMENTAL DESIGN

## A. OVERVIEW

The goal of this research is to implement the SNMP services on handheld devices with 802.11 network capabilities and test the performance on several commercially available devices. The initial objective for the experiment is to make the SNMP proxy agent inside Pocket PC detectable on the SNMP manger. In the next step, the SNMP manager can pull out the requested data from the MIB inside the application. In addition, the experiment includes the performance testing of SNMP manager to communicate with the SNMP agent over the wireless network. In the realistic working environment Pocket PC is turn on and off occasionally, therefore it is a significant factor to test the duration of the time that SNMP managers can discover the mobile devices on the wireless environment. The SNMP Proxy agent application is written with Embedded Visual Basic using an ActiveX Control from IP*Works! . This experiment used the two wireless Pocket PCs equipped with the Intel X-Scale Processor: Toshiba E740, and HP IPAQ 5450.

## B. HARDWARE ARCHITECTURE

An experimental network was created on Ethernet topology with three desktop PCs consisting of one server and two clients (see Figure 8). The server runs Microsoft Windows 2000 Server; both clients run Microsoft Windows 2000 Professional and Microsoft Windows XP Professional. All devices are connected by a NETGEAR dual 10/100 speed 6 port hub. One port of the internal hub is assigned to a Linksys WAP11 Access Point (AP), connected to a wireless client using the same subnet as the wired network. The wireless mobile devices consist of HP IPAQ 5450 and Toshiba E740. Both communicate with the network by an 802.11b wireless access point.

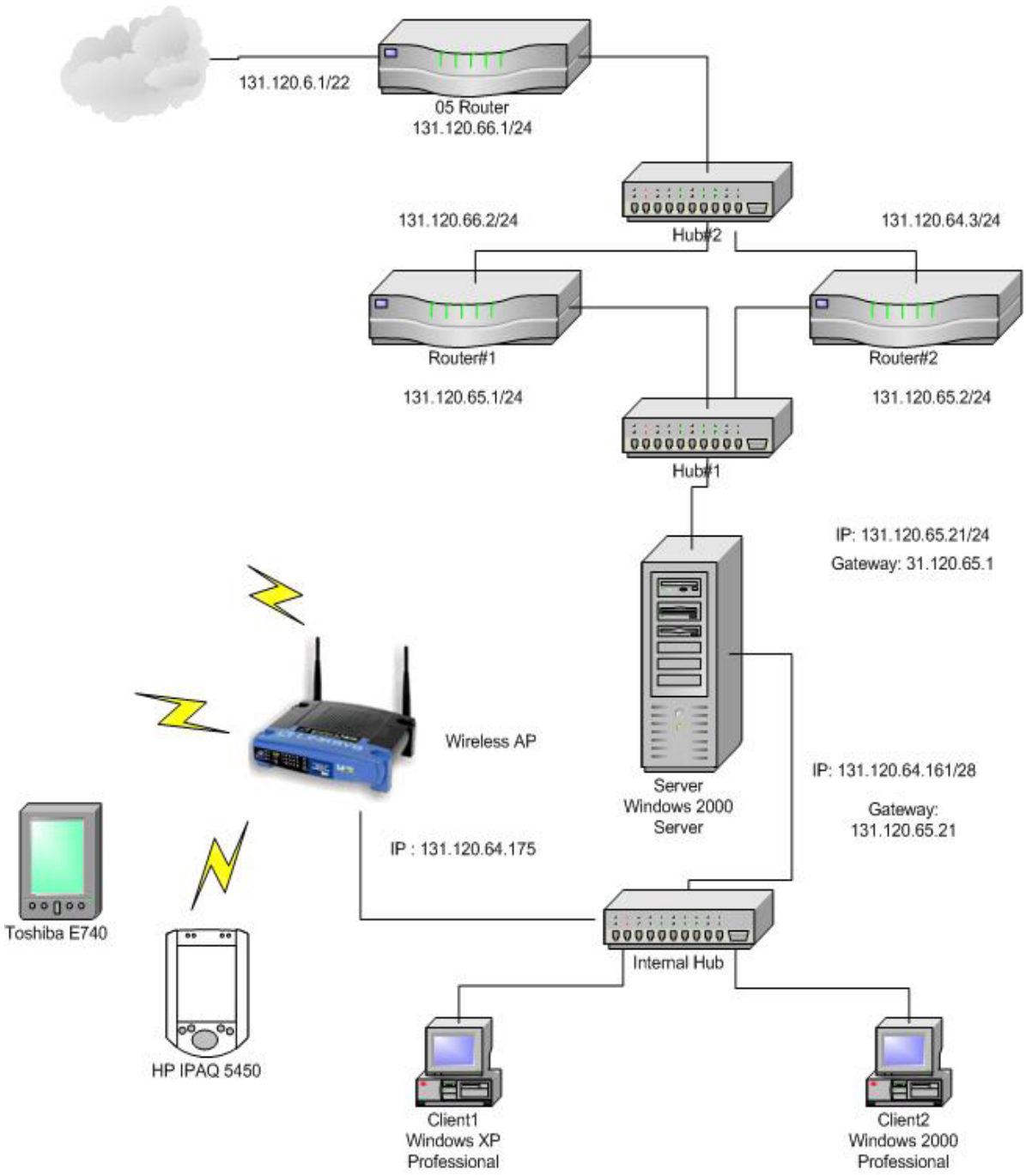



**Figure 9 – Experimental network architecture**

## C. SOFTWARE ARCHITECTURE

### 1. Architecture Overview

The Server, running Windows 2000 Server, is installed with three different SNMP managers. One Windows XP professional and One Windows 2000 Professional client is running a native SNMP agent. The portable wireless clients are tested with the proxy SNMP agent installed as shown in Figure 10.

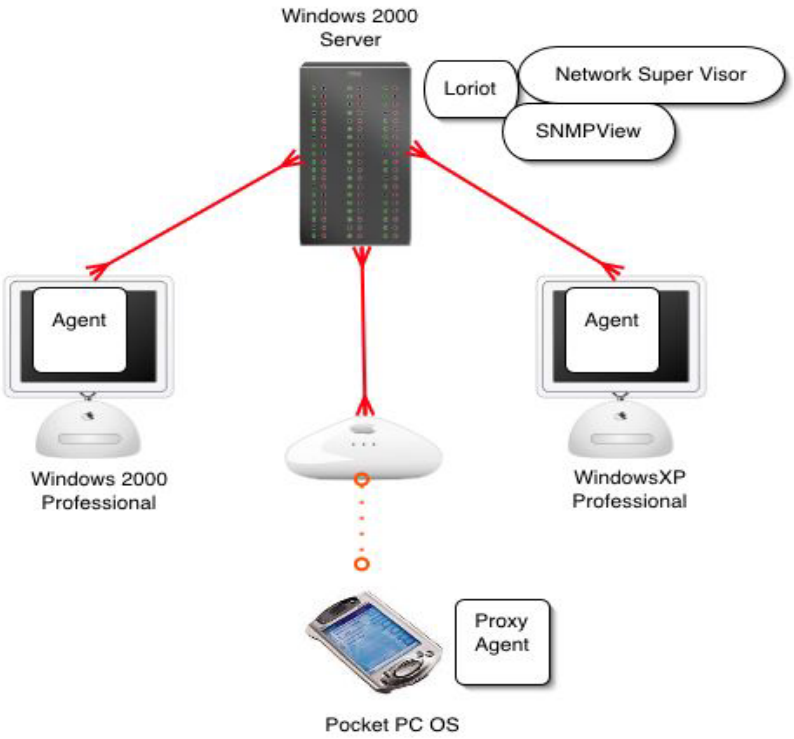



**Figure 10 - The Software Architecture**

## 2. The SNMP Agent

### *a. SNMP Manageable Devices*

In order to use SNMP communication on a Windows CE client, both PC clients must enable the SNMP service. Microsoft has an SNMP agent in Windows XP, 2000 (Professional) and 98. The system can be tested by using the local loop back address of 127.0.0.1 to access the agent even if the systems are not connected to a physical network. The step-by-step process to enable the SNMP service is described in Appendix C.

### *b. Non-SNMP Manageable Devices*

For non-SNMP manageable devices such as Windows 95 and Windows CE including Pocket PC, an SNMP Proxy agent must be implemented. A proxy agent in this experiment is written in EVB by using Microsoft Visual Tool. The frameworks for the application are .NET framework 1.1 for desktop and ActiveX control for Pocket PC 2002: both frameworks are from IP*Works!

There are many approaches available to implement the SNMP proxy agent for the system. A lot of supported tools and packages exist for almost every programming language. However the majority of SNMP agent developments are focused on C++ and Java. Typically C++ is the best choice for Windows application developer because of the compatibility with Windows operating system. In addition Java is one of the good alternatives to implement the system that can run on many platforms.

Nevertheless there are only a few alternative approaches to implement an application on a limited device such as C++, Java, Visual Basic and C#. For C++ developers, SNMP++ and Agent++ (http://www.agentpp.com) are the completed tool sets to implement an SNMP agent on Pocket PC because a Germany company has released the subagent for Windows CE in early March 2003. Therefore a developer can use subagent for SNMP stack in SNMP++ and Agent++ library to implement the SNMP application.

For Java, there are a lot of SNMP packages that developer can use such as a product form Advent Network Management [16] or Java Management Extensions (JMX), ( http://java.sun.com/products/JavaManagement) from Sun Microsystems. Although Java is an excellent initiative for a Pocket PC programming, most of the SNMP Java classes are relatively big and run extremely slow on the Pocket PC operating system.

Although C# and .NET Compact Framework seems to be the best option, Microsoft did not release the final version of Microsoft Visual Studio .NET 2003 yet. In additional Microsoft Visual Studio .NET 2003 still has a problem with the connected devices as the result it is not a good choice for Pocket PC development yet. Therefore an Embedded Visual Basic is a best choice for now because of its compatible with Pocket PC and ActiveX control.

IP*Works! ActiveX Edition currently supports Pocket PC 2002, Pocket PC 2000, HPC 2000, and Smart Phone. These products provide the same interfaces and the same functionality as the corresponding desktop versions, but in an entirely different form factor. IP*Works! for Windows CE - including Pocket PC (Windows CE V3.0) - allows developer to build applications with Internet communications functionality including SNMP for network management. The IP*Works! for Windows CE includes support for the Pocket PC emulator and inclusion of Common Executable Format (CEF) binaries. The ActiveX edition package contains 32 bit ActiveX Controls optimized for the Internet. The controls are small and efficient, with no dependencies on external libraries. They can be used from Visual Basic, and all other environments that support ActiveX Control such as MS Access, FoxPro, etc. The compatibility of ActiveX control is shown in Table 1.

[16] http://www.nemoto.ecei.tohoku.ac.jp/~nitou/snmpdocs/tutorial.html

| Operating System for Deployment |
| --- |
| • Windows CE 3.0 |
| Architecture of Product |
| • 32Bit |
| Tool Type |
| • Component |
| Component Type |
| • ActiveX (OCX) |
| Built Using |
| • ActiveX Template Library (ATL) |
| Business Function |
| • Internet Communication Components |
| General |
| Internet Enhanced<br>Includes Digital Signature<br>Marked as Safe for Scripting<br>Marked as Safe for Initialization<br>OCX96 Compatible<br>Supports Apartment Model Threading<br>Microsoft Transaction Server Compatible (MTS) |

**Table 1 – ActiveX Compatibility From: Ref.15.**

## D. EXPERIMENTAL STRUCTURE

### 1. Experimental Procedure

- Enable the SNMP service on PC clients both Windows 2000 and Windows XP Professional.
- Test whether SNMP agent on clients can communicate with all three SNMP manager; Loriot, SNMPView, and 3COM Network Supervisor.
- Test the implemented proxy on the Windows desktop environment.
- Shutdown the SNMP service on Windows Clients, so an application can use port 161 and 162.
- Check if the SNMP managers are able to detect the Windows devices that turn off the SNMP service.
- Install and run the SNMP agent proxy application on Windows Clients.
- Test if the SNMP proxy agent is visible on SNMP managers.
- Test whether Pocket PCs are visible in the SNMP manager after starting an SNMP agent on the Pocket PC device.

### 2. Hypothesis

It is feasible to implement SNMP services software on a handheld wireless device. If the SNMP proxy agent, implemented on the desktop is successfully running and visible on SNMP agent, the SNMP proxy agent on Pocket PC should be observed by the SNMP manager as well because the same code for Pocket PC and desktop versions are reused with the same common framework from IP*Works! If the applications use the available limited features that work on both frameworks, both applications should get the same result.

## E. IMPLEMENTATION

The SNMP Proxy agent application is implemented with the architecture described on the Figure 10 by using the SNMP Component from IP*Works!, nsoftware.IPWorks.SNMP as described in Appendix E.

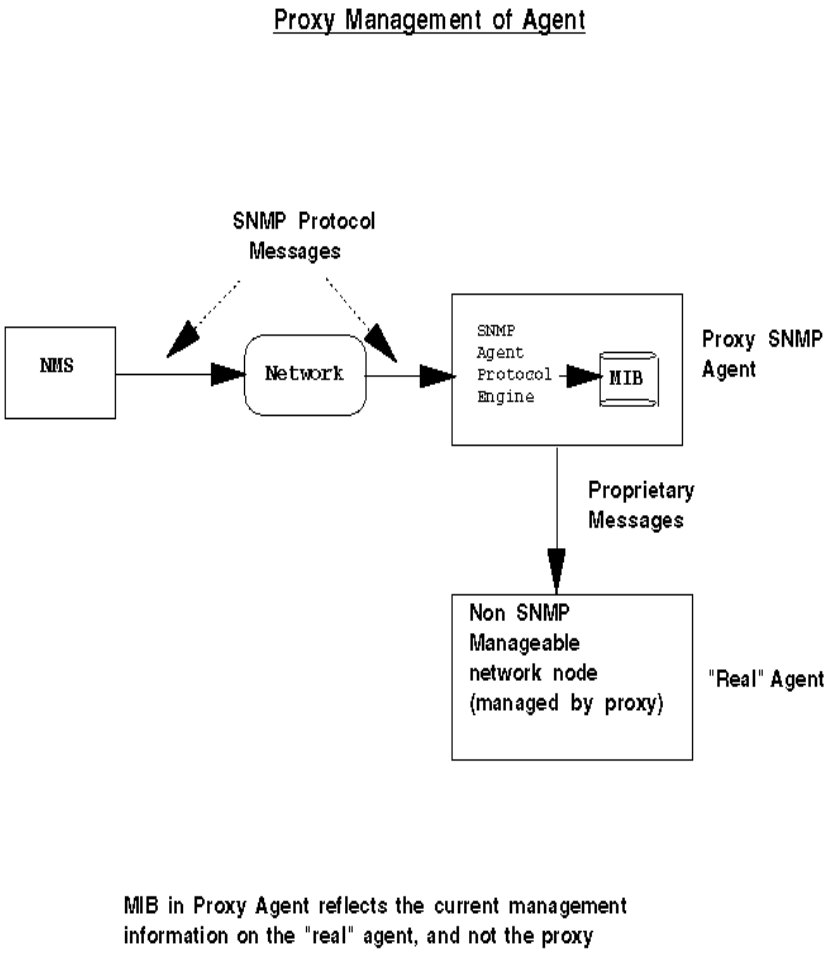



**Figure 11 – SNMP Proxy Application Architecture [From: Ref.02.]**

The SNMP component from IP*Works! implements a standard SNMP Agent as specified in the SNMP RFCs. both SNMP v1 and SNMP v2c . The component provides both encoding/decoding and transport capabilities. The SNMP agent is activated by setting the local port to 161. SNMP object ids, types, and values are provided in arrays.

The SNMP proxy agent application also includes the Trap command to send the signal to the manager. SNMP Traps are received through the Trap event and sent through the Action property by specifying appropriate values in the various Trap properties.

The MIB in an application is written on the top of ActiveX component by specifying an ObjID to each object such as system description has an id 1.3.6.1.2.1.1.1.0. The entire MIB objects for this application are kept in an array consists of seven element properties therefore an NMS can query the state of MIB by index. The SNMP Proxy agent application contains the following fields as shown in Figure 12 the screen shot from proxy agent prototype and Figure 13 the screen shot from Pocket Agent.

- sysDescr – the specification of the system.
    - MIB ID 1.3.6.1.2.1.1.1.0
- sysObjectID – the unique ID of the system.
    - MIB ID 1.3.6.1.2.1.1.2.0
- sysContact – the contact information.
    - MIB ID 1.3.6.1.2.1.1.3.0
- sysName – the name of the system.
    - MIB ID 1.3.6.1.2.1.1.4.0
- sysLocation –the location of the system.
    - MIB ID 1.3.6.1.2.1.1.5.0
- sysUptime – the duration since the agent start.
    - MIB ID 1.3.6.1.2.1.1.6.0
- sysService – define the service currently requested from agent: default is 0
    - MIB ID 1.3.6.1.2.1.1.7.0

The last part of the application interface includes the Trap command so user can input an IP address in the Remote Host field and be able to send Trap to a managed station.

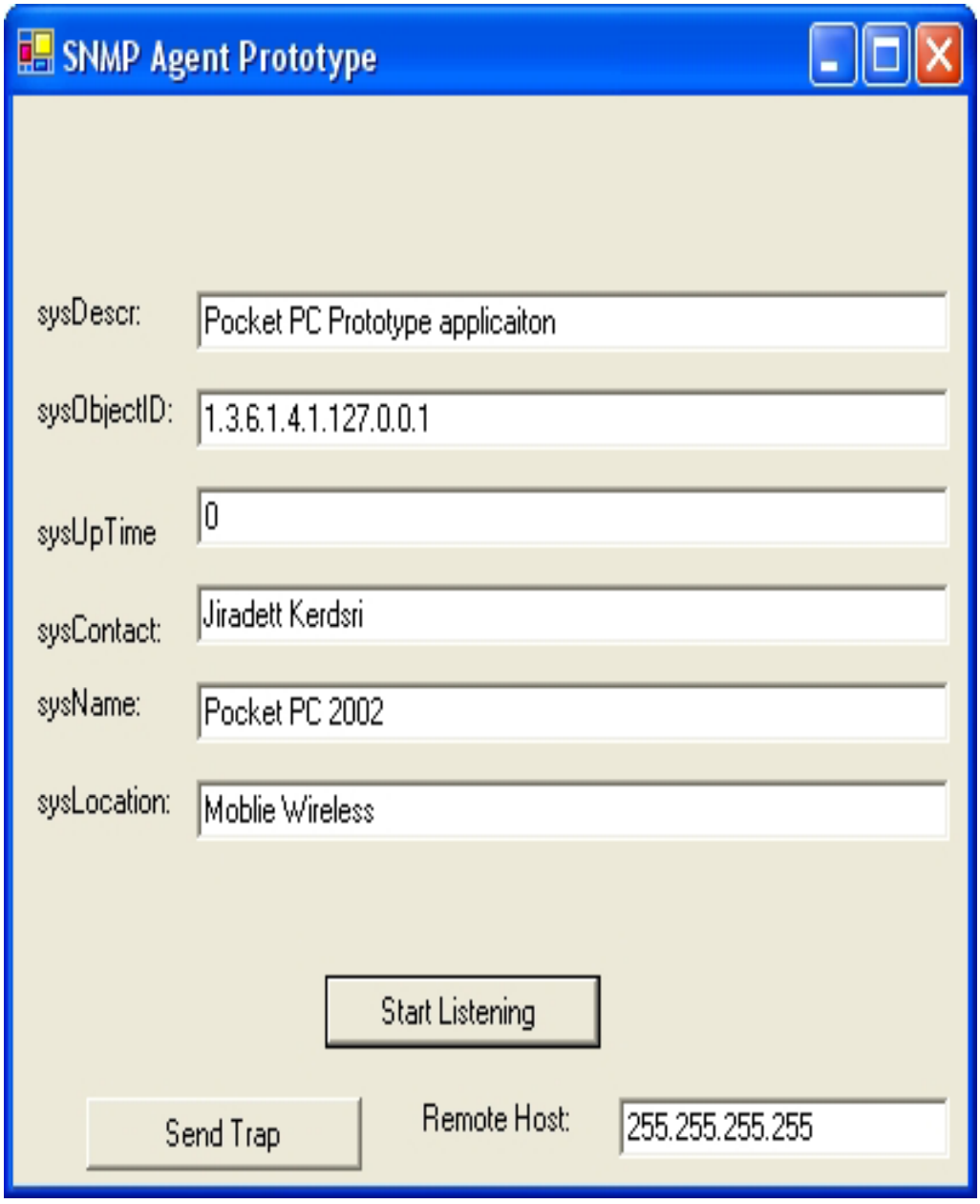



**Figure 12 – The Pocket Agent Prototype for desktop.**


**Figure 13 – The Pocket Agent for Pocket PC.**

## F. EXPERIMENTAL TESTING

### *a. SNMP Agent Prototype Testing*

All the network devices are tested for the SNMP communication by three SNMP mangers as mentioned in Chapter IV-C. Both PC Clients are connected to the server by specify IP address to the network management station. All managers are able to ping SNMP and discover the agent on both PC clients. Then an SNMP Agent Prototype running on the third device has been added to the system.

3COM Network Supervisor automatically detects all the devices and can view the entire SNMP devices graphically including an SNMP Proxy Agent although it can not get the name from the proxy agent as in Figure 14 because the NMS can not query the requested MIB from an agent. Network Supervisor can also view the properties of the entire devices except the proxy agent.

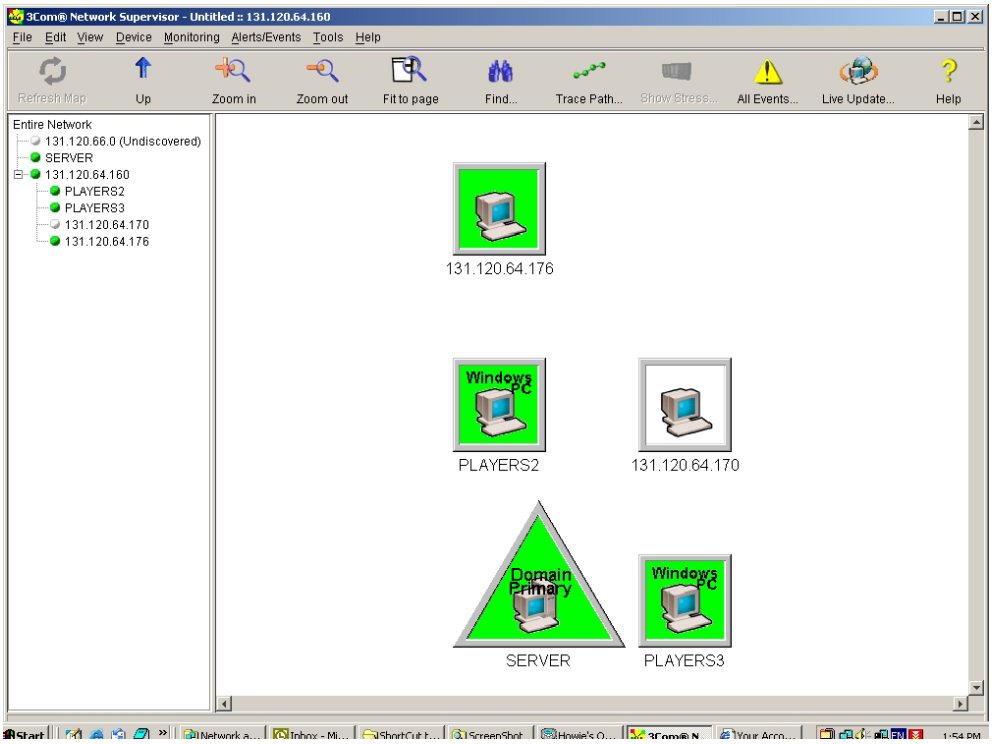



**Figure 14 – 3COM Network Supervisor.**

The Loriot can discover all the devices that connected to the NMS manually by specifying the IP addresses. All the devices are successfully connected to the Loriot as shown in Figure 15. The NMS can also query the system properties except for the proxy agent because an NMS cannot complete the requested command to an agent. The Loriot provide more detail than Network Supervisor as in Figure 16. Even though Loriot cannot query the MIBS from SNMP Proxy Agent, the Loriot can send the SNMP Ping to the proxy agent and receive the SNMP response back from SNMP Proxy Agent with the name that is specified in an application as shown in Figure 17.

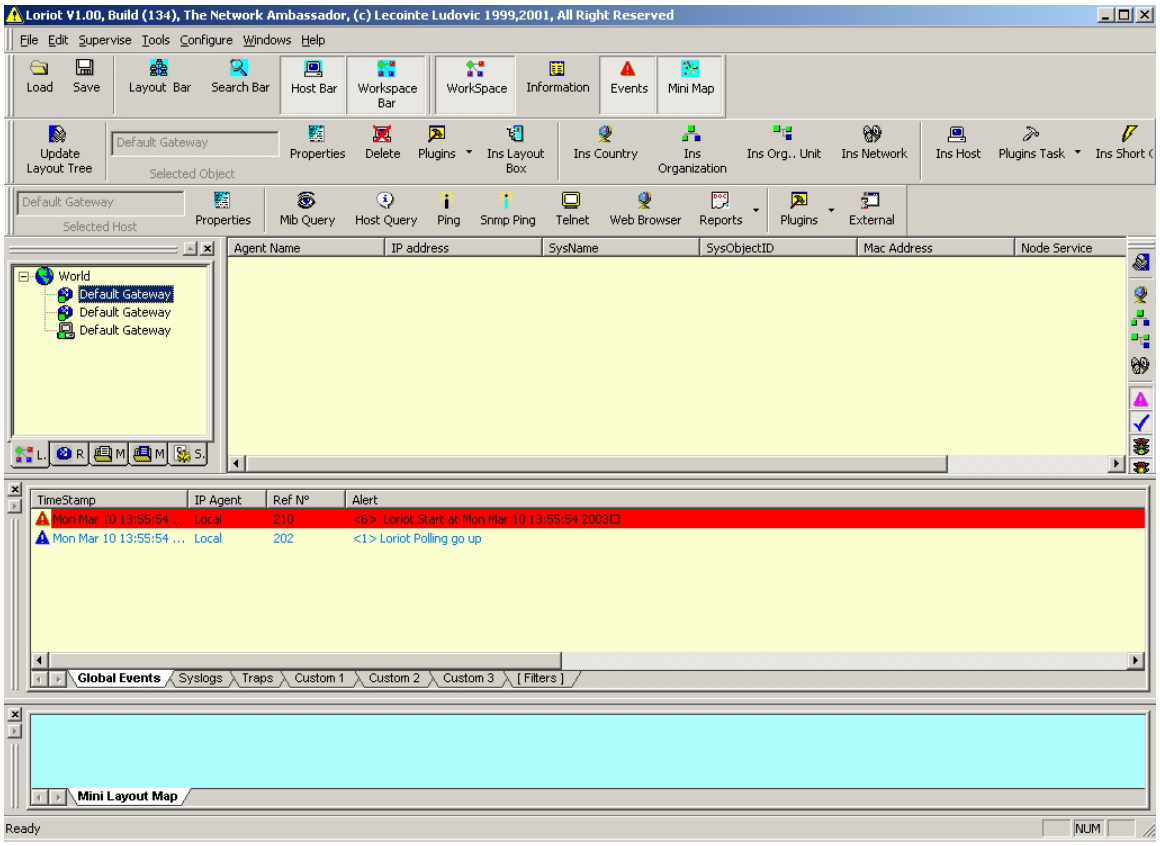



**Figure 15 – Loriot.**

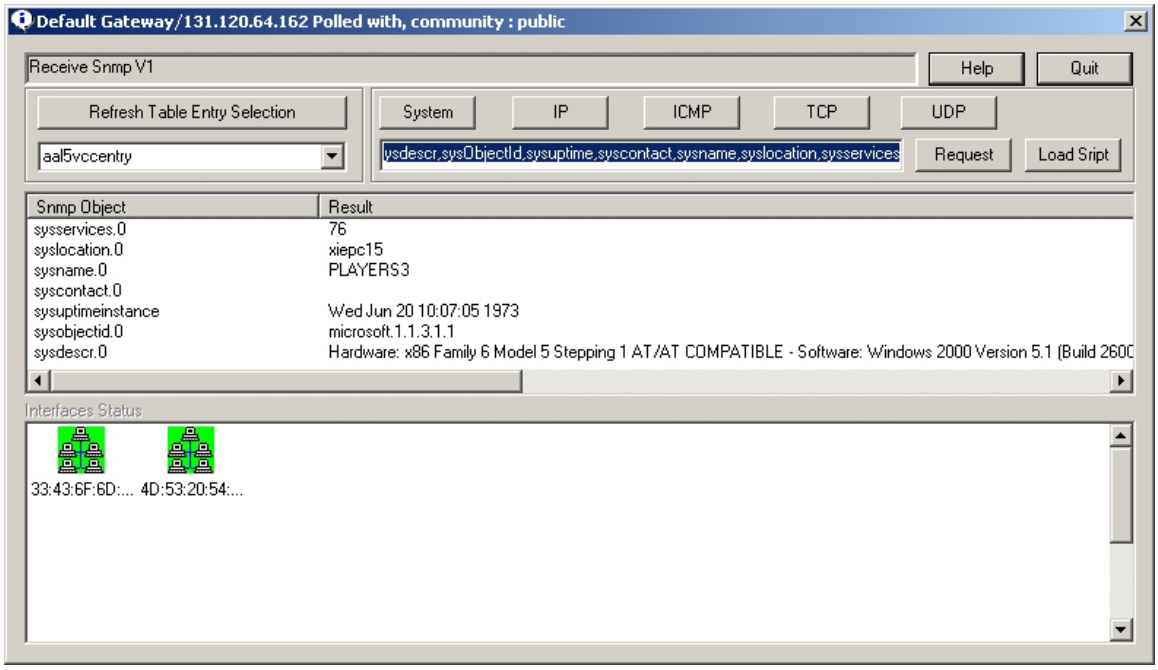



**Figure 16 – MIBS query in Loriot.**

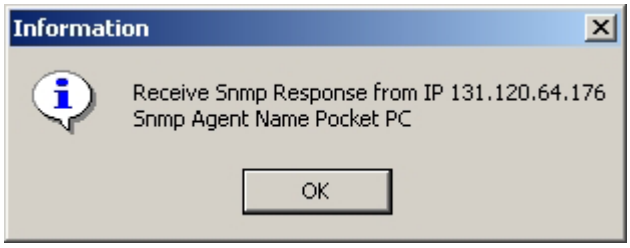



**Figure 17 – SNMP Ping for Proxy Agent.**

The last tested manager appears to be the most compatible to the implemented application because the SNMPView can query all the MIBs information from a proxy agent. Although SNMPView does not include the graphical detail, the SNMPView output is able to view the proxy agent's MIB over the web browser via port 8888. SNMPView can show all the MIB specifying in config file. There are 4 MIBs

requested for this test: System Location, System Description, System Name and System Uptime. All the devices communicated with SNMPView are successfully query the MIBs state from an NSM as shown in Figure 18.

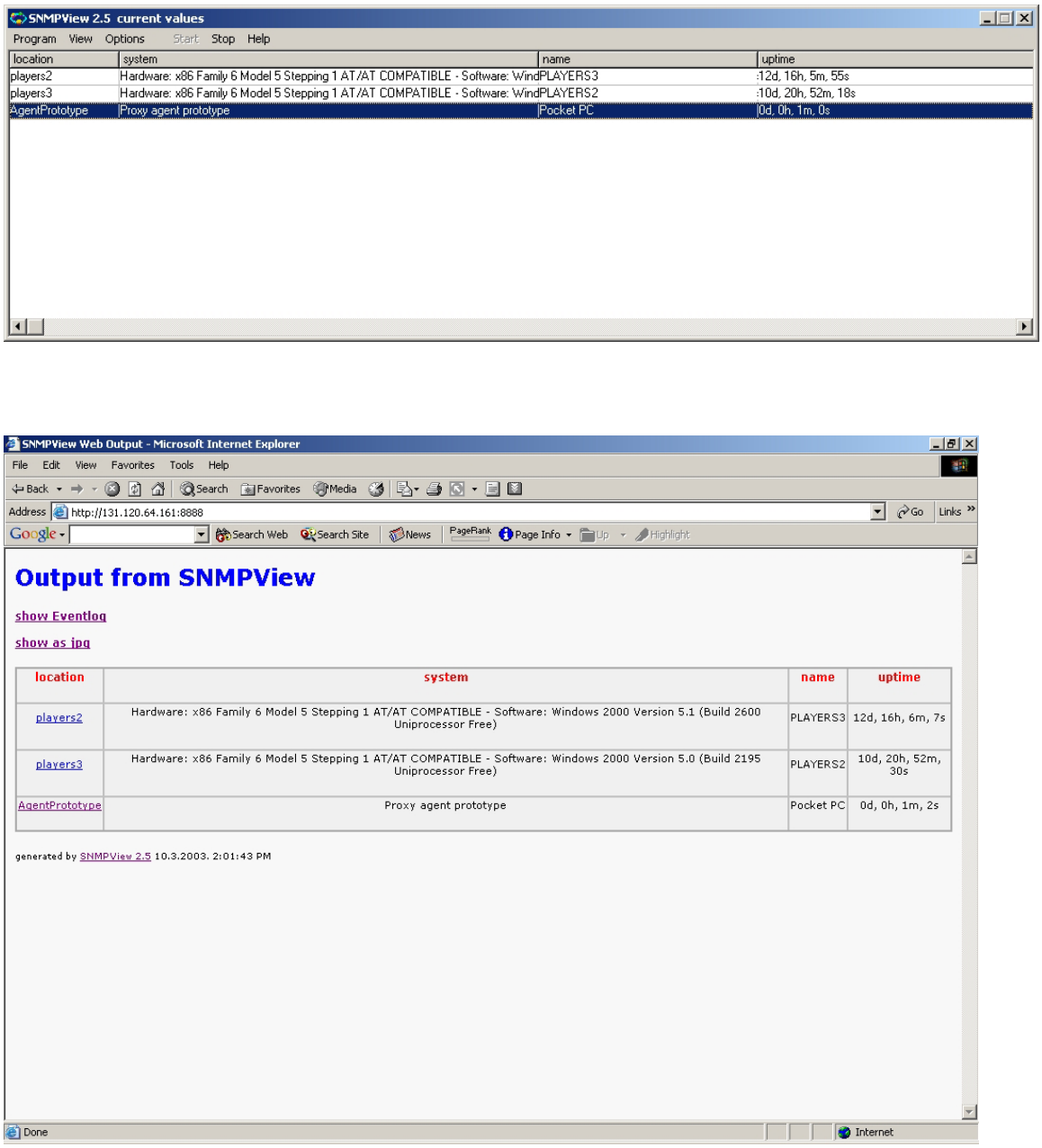



**Figure 18 – SNMPView.**

### *b. SNMP Agent Prototype Testing Result*

The tables below conclude the testing result for SNMP service on Windows system and Proxy Agent Prototype. The result conducts with 4 major features for SNMP Manager; SNMP Agent Visibility, Ping SNMP, MIBs Query,and Trap detecting.

- SNMP Agent Visibility is the ability to detect the SNMP agent in the local subnet both automate and manual.
- Ping SNMP is the ability that SNMP can ping the SNMP service on an Agent device. This feature is available in the Loriot only.
- MIBs Query is the ability to acquire the system properties from an Agent device.
- Trap detecting is the ability for SNMP Manager to sense the Trap command from SNMP Agent.

Remark: In the tables Y = Yes , N = No, and N/A = Not Available

| NMS/Feature Testing | SNMP Visibility | Ping SNMP | MIBs Query | Trap detecting |
|---|---|---|---|---|
| Network Supervisor | Y | N/A | Y | N/A |
| Loriot | Y | Y | Y | Y |
| SNMPView | Y | N/A | Y | Y |

**Table 2 – Compatibility test result from native agent.**

| NMS/Feature Testing | SNMP Visibility | Ping SNMP | MIBs Query | Trap detecting |
|---|---|---|---|---|
| Network Supervisor | Y | N/A | N | N/A |

| | | | | |
|---|---|---|---|---|
| Loriot | Y | Y | N | Y |
| SNMPView | Y | N/A | Y | N |

**Table 3 – Compatibility test result from proxy agent.**

### *c. PocketAgent Testing*

An SNMP proxy agent for Pocket PC is written on Embedded Visual Basic, which has a Pocket PC 2002 Emulator, integrated. Therefore an application can be tested on an Emulator before connected to a real device. However the ActiveX control component from IP*Works! has a connection problem with x86 platform Emulator. As the result, all the Pocket PC experiments are conducted on the Pocket PC devices.

An SNMP proxy agent on PC is translated in to Pocket PC edition. Then an SNMP agent Pocket PC version is packed to a CAB file to install in to Pocket PC. The Pocket PCs is connected to the wireless network, and start an SNMP agent application.

The 3COM Network Supervisor can automatically discover the Pocket PC agent within the local subnet. An NMS then can monitor the status of the connected devices as shown in Figure 19.

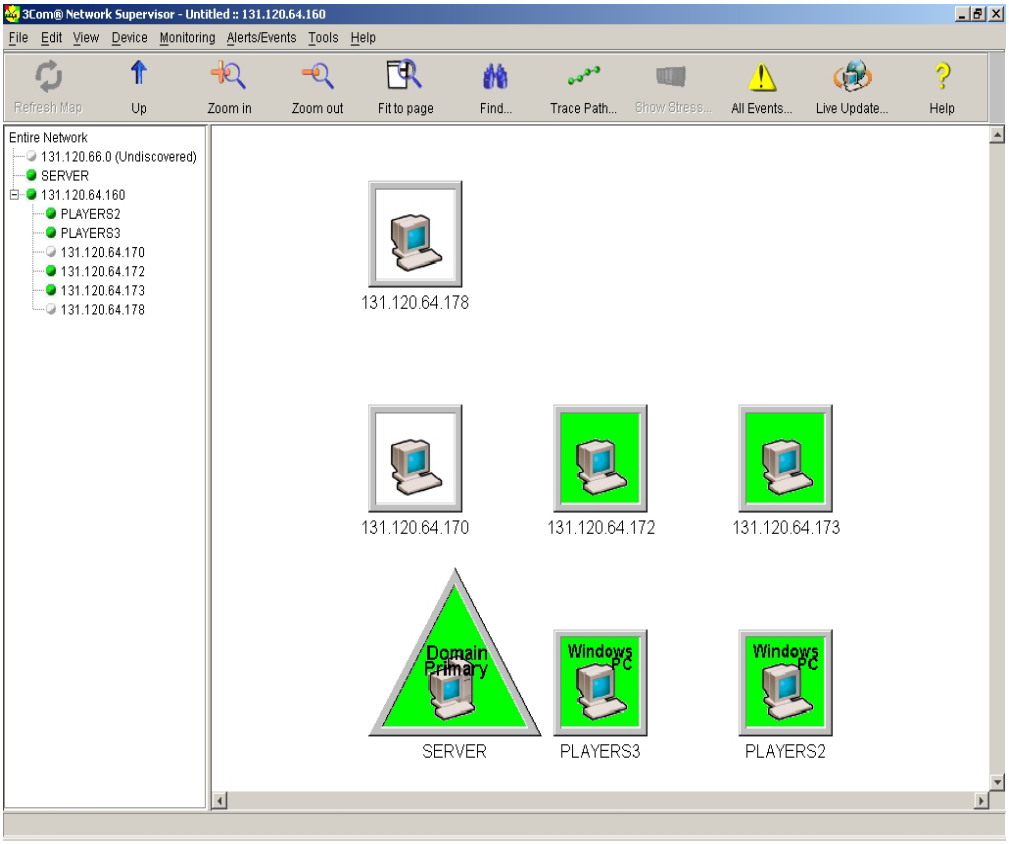



**Figure 19 – 3COM Network Supervisor Pocket PC.**

The SNMP agent Pocket PC can be added to the Loriot manually by specify IP address. The NMS can ping the SNMP agent on the Pocket PC device as shown in Figure 20 and 21. However the MIBs from Pocket Agent can not complete the request from NMS. The SNMP agent can successfully send Trap command to the manager. However a Loriot cannot query the MIB properties from an agent.

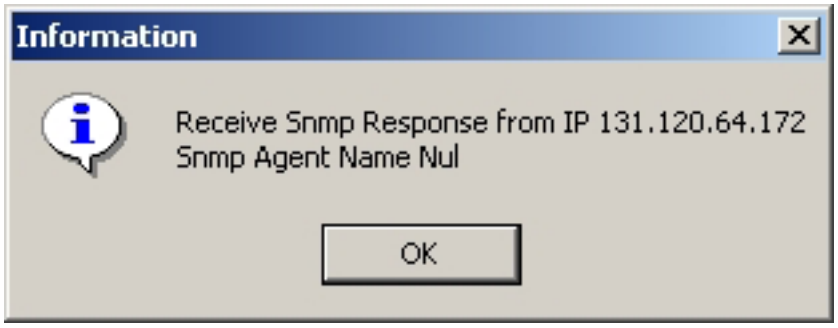



**Figure 20 – SNMP Ping response from Loriot.**

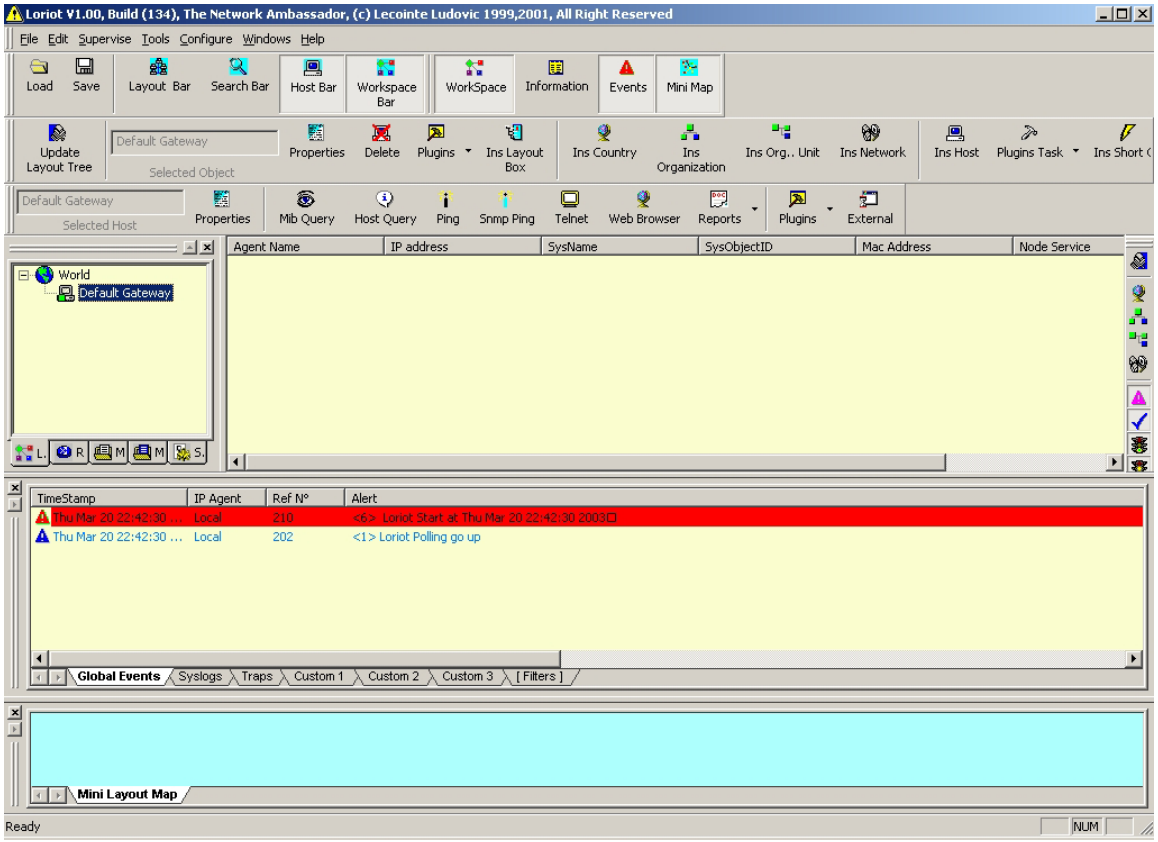



**Figure 21 – Added device in Loriot.**

An SNMPView cannot detect SNMP agent on Pocket PC because the problem with SNMP standard therefore an SNMP Manger created using IP*Works! component has been used to test the SNMP agent because an SNMP agent is not fully compatible with the tested SNMP managers. An SNMP manager from IP*Works! is able to automatically discover an SNMP agent, moreover it can query the MIB properties in Pocket PC as shown in Figure 21. An agent can also send Trap command to the manger by input the remote host in Pocket PC agent. The testing result is shown in Table 4.

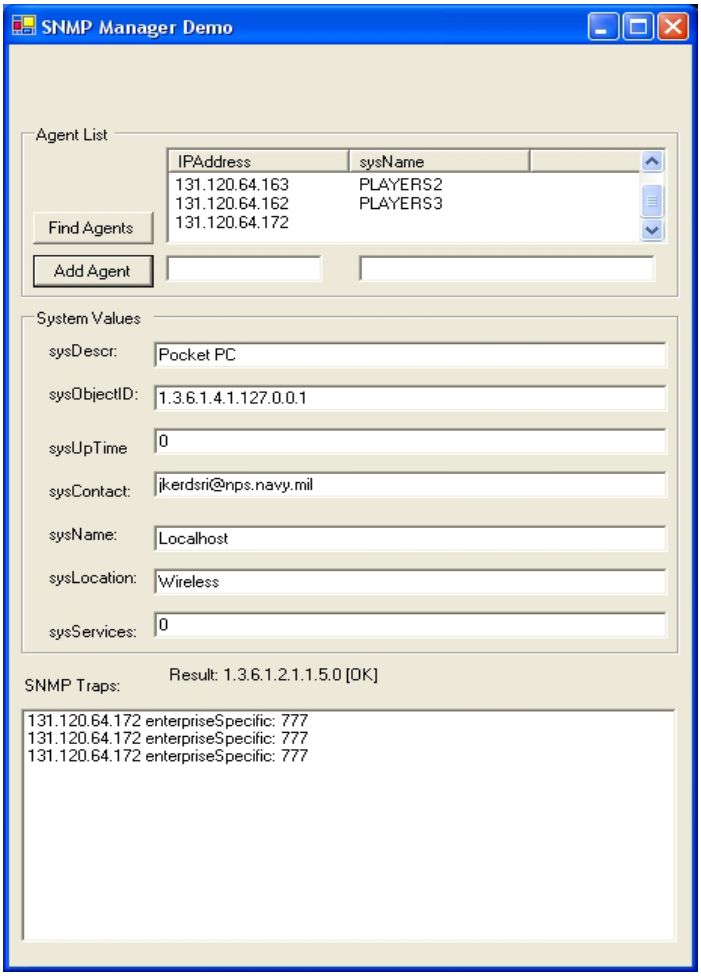



**Figure 22 – SNMP Manger from IP*Works!**

| NMS/Feature Testing | SNMP Visibility | Ping SNMP | MIBs Query | Trap detecting |
|---|---|---|---|---|
| Network Supervisor | Y | N/A | N | N/A |
| Loriot | Y | Y | N | Y |
| SNMPView | N | N/A | N | N |
| IP*Works! | Y | N/A | Y | Y |

**Table 4 – An SNMP proxy agent Pocket PC test result.**

## G. SUMMARY

This chapter describes the experimental design and testing result. An SNMP proxy agent running on desktop PC seems to get the same result as the standard SNMP service on Windows 2000 and XP Professional. However after migrating an SNMP agent from desktop PC to Pocket PC version, the SNMP agent capabilities were reduced due to the limitation of framework and device because the component in Windows CE development is the subset of Windows development framework. As the result some component in Windows CE edition cannot use full functionality as Windows desktop. More over the performance of the Pocket PC is less than desktop PC in the aspect of speed, wireless signal and delay.

THIS PAGE INTENTIONALLY LEFT BLANK

# V. CONCLUSION

## A. SUMMARY

Many problems arise in wireless mobile devices due to the communication between SNMP applications that rely on different platforms. Some platforms are proprietary solutions that do not comply with the open industry standards and cannot interoperate with any network device other than their own products. This is a problem for network managers because the network may include equipment and software from different vendors that do not subscribe to the same proprietary platform. These devices cannot be managed by the management system; the manager is "blind" to these devices' activity and performance. The proxy SNMP agent that conforms to the open standard must be implemented in order to communicate with the standard SNMP manager.

## B. CASE STUDY

In the past there were some research explained how to solve the problem between a Pocket PC and SNMP. One of the solutions offered is Embedded Linux. Several SNMP applications are available for Linux platform such as NET-SNMP, making the developers' job easier when implementing an SNMP agent for the Linux and using the Embedded Linux on a Pocket PC. Even though somewhat impractical having Linux OS on the Pocket PC device, it is easier to format Pocket PC OS to Linux rather than to implement it, such as a SNMP agent for Pocket PC. This case study is ideal for the developers familiarize with Linux OS and for the organizations using the Linux platform and requiring the portable device to communicate with the server.

## C. FUTURE RESEARCH

While this thesis concentrated on the possibility of using Microsoft Visual Tool and IP*Works! Framework, other areas of programming and tools are available that can implement the system. However IP*Works! is a complete tool set for implementing an internet application on many platforms. The first alternative solution is C# and Microsoft .NET Compact Framework that seems to be the most compatible and practical for Windows software implementation; therefore, a C# version of SNMP agent and SNMP manger should be implemented on .NET framework and .NET compact framework. This

results in a full SNMP system that can be set up on any device that supports .NET strategies, such as Windows CE or SmartPhone following the standard. Java is the other alternative area that should be implemented and tested for the SNMP application because Java is a versatile and flexible language that can run on the other portable device platforms, such as Palm and Symbian.

A MIB query feature in this research application is not fully functional and needs to be addressed to conform with the open standard. Moreover an SNMP agent should run in the background of the Pocket PC so that the agent will start up every time the user turns on the system.

In addition, the Pocket PC competitor, Palm OS, the most popular personal digital assistants, is the other area to be studied. The most difficult and challenging part of implementing such an SNMP system on a Palm device is the size of the application and available SNMP package to use. On the other hand, other embedded operating systems such as Symbian, Orange or Embedded Linux, are still the other areas that should be studied.

# APPENDIX A. SOURCE CODE

Appendix A consists of two source files: ProxyAgentPrototype.cs is written in C# and PocketAent.vb in Embedded Visual Basic.

ProxyAgentPrototype.cs

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
namespace SnmpAgent
{
/// <summary>
/// Summary description for Form1.
/// </summary>
public class SnmpAgent : System.Windows.Forms.Form
{
internal System.Windows.Forms.Button cmdSendTrap;
internal System.Windows.Forms.Button cmdListen;
internal System.Windows.Forms.Label Label6;
internal System.Windows.Forms.Label Label8;
internal System.Windows.Forms.Label Label9;
internal System.Windows.Forms.Label Label5;
internal System.Windows.Forms.Label Label4;
internal System.Windows.Forms.Label Label3;
internal System.Windows.Forms.Label Label2;
internal System.Windows.Forms.TextBox txtRemHost;
internal System.Windows.Forms.TextBox txtOIDVal5;
internal System.Windows.Forms.TextBox txtOIDVal4;
internal System.Windows.Forms.TextBox txtOIDVal3;
internal System.Windows.Forms.TextBox txtOIDVal2;
internal System.Windows.Forms.TextBox txtOIDVal1;
internal System.Windows.Forms.TextBox txtOIDVal0;
private nsoftware.IPWorks.Snmp snmp1;
private System.Timers.Timer timer1;
private int sysUpTime = 0;
/// <summary>
/// Required designer variable.
/// </summary>
private System.ComponentModel.Container components = null;
public SnmpAgent()
```

```
{
//
// Required for Windows Form Designer support
//
InitializeComponent();
//
// TODO: Add any constructor code after InitializeComponent call
//
snmp1.LocalPort = 161; // act as agent
}
/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}

Windows Form Designer generated code
/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
Application.Run(new SnmpAgent());
}
private void cmdListen_Click(object sender, System.EventArgs e)
{
if (snmp1.Active) {
sysUpTime = 0;
txtOIDVal2.Text = sysUpTime.ToString();
} else {
snmp1.RemoteHost = txtRemHost.Text;
}
snmp1.Active = !snmp1.Active;
timer1.Enabled = snmp1.Active;
cmdListen.Text = snmp1.Active ? "Stop Listening" : "Start Listening";
```

```
}
private void cmdSendTrap_Click(object sender, System.EventArgs e)
{
bool initialState = snmp1.Active;
snmp1.Reset();
snmp1.TrapEnterprise = txtOIDVal1.Text;
snmp1.TrapGenericType = nsoftware.IPWorks.SnmpTrapGenericTypes.
tgtEnterpriseSpecific;
snmp1.TrapSpecificType = 777;
snmp1.TrapTimeStamp = System.Int32.Parse(txtOIDVal2.Text);
snmp1.SendTrap();
if (!initialState) snmp1.Active = false;
}
private void timer1_Elapsed(object sender, System.Timers.ElapsedEventArgs e)
{
sysUpTime += 1;
txtOIDVal2.Text = sysUpTime.ToString();
}
private void snmp1_OnGetRequest(object sender, nsoftware.IPWorks.
SnmpGetRequestEventArgs e)
{
snmp1.RemoteHost = e.SourceAddress;
snmp1.RemotePort = e.SourcePort;
for (int i = 1; i <= snmp1.ObjCount; i++){
SetOIDVals(GetIndex(snmp1.ObjId[i]), i);
}
snmp1.SendGetResponse();
snmp1.Reset();
}
private void snmp1_OnGetNextRequest(object sender, nsoftware.IPWorks.
SnmpGetNextRequestEventArgs e)
{
snmp1.RemoteHost = e.SourceAddress;
snmp1.RemotePort = e.SourcePort;
int OIDidx = GetIndex(snmp1.ObjId[1]) + 1;
if (OIDidx >= 0 && OIDidx < 8){ //within OID value range
snmp1.ObjId[1] = "1.3.6.1.2.1.1." + OIDidx.ToString() + ".0";
SetOIDVals(OIDidx, 1);
snmp1.SendGetResponse();
snmp1.Reset();
}
}
private void snmp1_OnSetRequest(object sender, nsoftware.IPWorks.
SnmpSetRequestEventArgs e)
{
```

```
for (int i = 1; i <= snmp1.ObjCount; i++)
{
int OIDidx = GetIndex(snmp1.ObjId[i]);
switch(OIDidx)
{
case 1: txtOIDVal0.Text = snmp1.ObjValue[i];break;
case 2: txtOIDVal1.Text = snmp1.ObjValue[i];break;
case 3: txtOIDVal2.Text = snmp1.ObjValue[i];break;
case 4: txtOIDVal3.Text = snmp1.ObjValue[i];break;
case 5: txtOIDVal4.Text = snmp1.ObjValue[i];break;
case 6: txtOIDVal5.Text = snmp1.ObjValue[i];break;
case 7: txtOIDVal6.Text = snmp1.ObjValue[i];break;
}
}
snmp1.Reset();
}
private int GetIndex(string OID)
{
switch(OID)
{
case "1.3.6.1.2.1.1.1.0": return 1;
case "1.3.6.1.2.1.1.2.0": return 2;
case "1.3.6.1.2.1.1.3.0": return 3;
case "1.3.6.1.2.1.1.4.0": return 4;
case "1.3.6.1.2.1.1.5.0": return 5;
case "1.3.6.1.2.1.1.6.0": return 6;
case "1.3.6.1.2.1.1.7.0": return 7;
default: return 0;
}
}
private void SetOIDVals(int OIDidx, int Index)
{
snmp1.Community = "public";
switch(OIDidx)
{
case 1: snmp1.ObjType[Index] = nsoftware.IPWorks.SnmpObjTypes.
otOctetString;break;
case 2: snmp1.ObjType[Index] = nsoftware.IPWorks.SnmpObjTypes.otObjectId;
break;
case 3: snmp1.ObjType[Index] =
nsoftware.IPWorks.SnmpObjTypes.otTimeTicks;
break;
case 4: snmp1.ObjType[Index] = nsoftware.IPWorks.SnmpObjTypes.
otOctetString;break;
case 5: snmp1.ObjType[Index] = nsoftware.IPWorks.SnmpObjTypes.
```

```
otOctetString;break;
case 6: snmp1.ObjType[Index] = nsoftware.IPWorks.SnmpObjTypes.
otOctetString;break;
case 7: snmp1.ObjType[Index] = nsoftware.IPWorks.SnmpObjTypes.otInteger;
break;
default: snmp1.ErrorStatus = 2; return;
}
switch(OIDidx)
{
case 1 : snmp1.ObjValue[Index] = txtOIDVal0.Text;break;
case 2 : snmp1.ObjValue[Index] = txtOIDVal1.Text;break;
case 3 : snmp1.ObjValue[Index] = txtOIDVal2.Text;break;
case 4 : snmp1.ObjValue[Index] = txtOIDVal3.Text;break;
case 5 : snmp1.ObjValue[Index] = txtOIDVal4.Text;break;
case 6 : snmp1.ObjValue[Index] = txtOIDVal5.Text;break;
case 7 : snmp1.ObjValue[Index] = txtOIDVal6.Text;break;
}
}
}
}
```

PocketAgent.vb

```
Form1 - 1
Option Explicit
Dim Listening As Boolean
Dim sysUpTime As Integer
Dim txtOIDVal(8) As TextBox
Private Sub Form_Load()
txtOIDVal0.Text = "Pocket PC"
End Sub
Private Sub cmdSendTrap_Click()
SNMP1.Reset
SNMP1.TrapEnterprise = txtOIDVal(1).Text
SNMP1.TrapGenericType = tgtEnterpriseSpecific
SNMP1.TrapSpecificType = 777
SNMP1.TrapTimeStamp = CLng(txtOIDVal(2).Text)
SNMP1.SendTrap
End Sub
Private Sub Timer1_Timer()
sysUpTime = sysUpTime + 1
txtOIDVal(2).Text = sysUpTime
End Sub
Private Sub SNMP1_GetNextRequest(RequestId As Long, Community As String,
SourceAddress As String, Sour
```

```
cePort As Integer)
Dim OIDidx As Integer
SNMP1.RemoteHost = SourceAddress
SNMP1.RemotePort = SourcePort
OIDidx = GetIndex(SNMP1.ObjId(1)) + 1
If OIDidx >= 0 And OIDidx < 8 Then
SNMP1.ObjId(1) = "1.3.6.1.2.1.1." & OIDidx & ".0"
SetOIDVals OIDidx, 1
SNMP1.SendGetResponse
SNMP1.Reset
End If
End Sub
Private Sub SNMP1_GetRequest(RequestId As Long, Community As String,
SourceAddress As String, SourcePo
rt As Integer)
Dim i As Integer
SNMP1.RemoteHost = SourceAddress
SNMP1.RemotePort = SourcePort
For i = 1 To SNMP1.ObjCount
SetOIDVals GetIndex(SNMP1.ObjId(i)), i
Next i
SNMP1.SendGetResponse
SNMP1.Reset
End Sub
Private Sub SNMP1_SetRequest(RequestId As Long, Community As String,
SourceAddress As String, SourcePo
rt As Integer)
Dim i, OIDidx As Integer
For i = 1 To SNMP1.ObjCount
OIDidx = GetIndex(SNMP1.ObjId(i))
If OIDidx > 0 Then txtOIDVal(OIDidx - 1).Text = SNMP1.ObjValue(i)
Next i
SNMP1.Reset
End Sub
Private Sub cmdListen_Click()
Form1 - 2
'Toggle listening, initialize
Listening = Not Listening
If Listening Then
SNMP1.LocalPort = 161
SNMP1.RemoteHost = tRemHost.Text
SNMP1.Active = True
cmdListen.Caption = "Stop Agent"
Timer1.Enabled = True
Else
```

```
SNMP1.Active = False
Timer1.Enabled = False
sysUpTime = 0
txtOIDVal(2).Text = sysUpTime
cmdListen.Caption = "Start Agent"
End If
End Sub
Private Sub SetOIDVals(ByVal OIDidx As String, Index As Integer)
SNMP1.Community = "public"
Select Case OIDidx
Case 1: SNMP1.ObjType(Index) = otOctetString
Case 2: SNMP1.ObjType(Index) = otObjectId
Case 3: SNMP1.ObjType(Index) = otTimeTicks
Case 4: SNMP1.ObjType(Index) = otOctetString
Case 5: SNMP1.ObjType(Index) = otOctetString
Case 6: SNMP1.ObjType(Index) = otOctetString
Case 7: SNMP1.ObjType(Index) = otInteger
Case Else:
SNMP1.ErrorStatus = 2
End Select
If SNMP1.ErrorStatus = 0 Then
SNMP1.ObjValue(Index) = txtOIDVal(OIDidx - 1).Text
End If
End Sub
Private Function GetIndex(ByVal OID As String) As Integer
Select Case OID
Case "1.3.6.1.2.1.1.1.0": GetIndex = 1
Case "1.3.6.1.2.1.1.2.0": GetIndex = 2
Case "1.3.6.1.2.1.1.3.0": GetIndex = 3
Case "1.3.6.1.2.1.1.4.0": GetIndex = 4
Case "1.3.6.1.2.1.1.5.0": GetIndex = 5
Case "1.3.6.1.2.1.1.6.0": GetIndex = 6
Case "1.3.6.1.2.1.1.7.0": GetIndex = 7
Case Else: GetIndex = 0
End Select
End Function
Form1 - 1
VERSION 5.00
Object = "{53337303-F789-11CE-86F8-0020AFD8C6DB}#1.0#0";
"snmp50.ocx"
Begin VB.Form Form1
Appearance = 0 'Flat
BackColor = &H80000005&
Caption = "PocketAgent"
ClientHeight = 3615
```

```
ClientLeft = 60
ClientTop = 840
ClientWidth = 3480
ForeColor = &H80000008&
ScaleHeight = 3615
ScaleWidth = 3480
ShowOK = -1 'True
Begin VBCE.Timer Timer1
Left = 360
Top = 0
_cx = 847
_cy = 847
Enabled = -1 'True
Interval = 0
End
Begin VBCE.CommandButton cmdSendTrap
Height = 255
Left = 120
TabIndex = 3
Top = 3360
Width = 3255
_cx = 5741
_cy = 450
BackColor = 12632256
Caption = "Send Trap"
Enabled = -1 'True
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Tahoma"
Size = 8.25
Charset = 0
Weight = 400
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Style = 0
End
Begin VBCE.TextBox tRemHost
Height = 255
Left = 1320
TabIndex = 4
Top = 3000
Width = 1815
_cx = 3201
_cy = 450
```

```
BackColor = -2147483643
BorderStyle = 1
Enabled = -1 'True
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Tahoma"
Size = 8.25
Charset = 0
Weight = 400
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
ForeColor = -2147483640
Text = "255.255.255.255"
Alignment = 0
HideSelection = -1 'True
Locked = 0 'False
MaxLength = 0
MultiLine = 0 'False
Form1 - 2
PasswordChar = ""
ScrollBars = 0
End
Begin VBCE.TextBox txtOIDVal
Height = 255
Left = 1320
TabIndex = 2
Top = 480
Width = 1815
_cx = 3201
_cy = 450
BackColor = -2147483643
BorderStyle = 1
Enabled = -1 'True
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Tahoma"
Size = 8.25
Charset = 0
Weight = 400
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
ForeColor = -2147483640
Text = "Pocket PC "
```

```
Alignment = 0
HideSelection = -1 'True
Locked = 0 'False
MaxLength = 0
MultiLine = 0 'False
PasswordChar = ""
ScrollBars = 0
End
Begin VBCE.Label Label1
Height = 255
Left = 0
TabIndex = 1
Top = 480
Width = 1095
_cx = 1931
_cy = 450
AutoSize = 0 'False
BackColor = -2147483643
BackStyle = 1
BorderStyle = 0
Caption = "sysDescr"
Enabled = -1 'True
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Tahoma"
Size = 8.25
Charset = 0
Weight = 400
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
ForeColor = -2147483640
Alignment = 0
UseMnemonic = -1 'True
WordWrap = 0 'False
End
Begin SNMPLibCtl.SNMP SNMP1
Left = 0
Top = 0
Community = "public"
InBufferSize = 2048
LocalPort = 0
OutBufferSize = 2048
RemoteHost = ""
RemotePort = 0
```

```
SNMPVersion = 1
Form1 - 3
WinsockLoaded = -1 'True
End
Begin VBCE.CommandButton cmdListen
Height = 285
Left = 600
TabIndex = 0
Top = 120
Width = 2745
_cx = 4842
_cy = 503
BackColor = 12632256
Caption = "Start Agent"
Enabled = -1 'True
BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
Name = "Tahoma"
Size = 8.25
Charset = 0
Weight = 400
Underline = 0 'False
Italic = 0 'False
Strikethrough = 0 'False
EndProperty
Style = 0
End
End
```

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX B. HARDWARE SPECIFICATION

| Personal Computer Specification | |
|---|---|
| Computer | **x86 Family 6 Model 5 Stepping 1 AT/AT compatible** |
| OS provided | **Microsoft Windows** |
| RAM | **261,672 KB** |
| Hard drive capacity | **4 GB** |
| CPU | **Pentium II Family** |
| Network Interface | **3Com EtherLink 10/100 PCI TX NIC (3c905B-TX) (on server)** |
| | **3Com EtherLink 10/100 PCI NIC (3c905-TX)** |
| Ports | **2 USB, PS2, Serial and Parallel** |

**Appendix B: Table 1 - Specification of Managed PCs**

| WAP11 Specification | |
|---|---|
| Device type | **Wireless access point** |
| Compatibility | **Windows** |
| Form factor | **External** |
| Connectivity technology | **Wireless** |
| Max range indoors / outdoors | **490 ft / 1640 ft** |
| Networking compliant standards | **IEEE 802.3-LAN, IEEE 802.11-LAN** |
| Data bandwidth | **2.4 GHz** |
| Data transfer rate | **11 Mbps** |
| Data link protocol | **Ethernet, Fast Ethernet** |

**Appendix B: Table 2 - Linksys WAP11 specification**

Toshiba E740 is the first Pocket PC in the market that equipped with Wi-Fi due to the popularity of the 802.11b standard. Toshiba is one of the good Pocket PC that packed

with all the features such as removable battery, built in compact flash slot, even though it has a main battery leak problem and the Toshiba Company has to recalled it back to fix.

| **Toshiba E740 Specification** | |
|---|---|
| Release date | **June 2002** |
| OS provided | **Microsoft Pocket PC 2002** |
| Installed RAM | **64 MB** |
| ROM installed (max) | **32 MB** |
| Weight | **6.3 oz** |
| Width | **0.6 in** |
| Depth | **4.9 in** |
| Height | **3.1 in** |
| Input device type | **Touch-screen** |
| Audio input | **Microphone** |
| Audio output | **Speaker(s)** |
| Processor | **Intel PXA250 400 MHz** |
| Modem | **No** |
| Built-in devices | **Speaker(s), microphone, display** |
| **Display** | |
| Display type | **3.5" color reflective TFT** |
| Color support | **16-bit (65K colors)** |
| Max resolution | **240 x 320** |
| **Expansion/Connectivity** | |
| Connectivity device(s) included | **Docking cradle** |
| Expansion slot(s) total (free) | **1 x CompactFlash Card, 1 x MultiMedia Card / SD Memory Card** |
| Port(s) total (free) / connector type | **1 x Wi-Fi, 1 x USB, 1 x headphone, 1 x microphone, 1 x AC adapter** |

| | |
|---|---|
| Wireless connectivity | **IrDA, Wi-Fi (IEEE 802.11b)** |

**Power**

| | |
|---|---|
| Battery installed (max) | **1 x Lithium Ion battery** |

### Appendix B: Table 3 - Toshiba E740 Specification

HP IPAQ 5450 is the most featured loaded Pocket PC that come with built in Wi-Fi and Bluetooth. As well as the finger print recognition so it is the solution for the large scale enterprise.

**HP IPAQ 5450 Specification**

| | |
|---|---|
| Release date | **November 2002** |
| OS provided | **Microsoft Pocket PC 2002** |
| Installed RAM | **64 MB** |
| ROM installed (max) | **48 MB Flash** |
| Weight | **7.1 oz** |
| Input device type | **Touch-screen** |
| Software included | **Calendar, Contacts, Task, Voice Recorder, Notes, Pocket Word, Pocket Excel, Pocket Internet Explorer, Windows Media Player 8, Calculator, Solitaire, Inbox, Microsoft Reader, File Explorer, MSN Messenger, Terminal Services Client, VPN Client** |
| Audio input | **Microphone** |
| Audio output | **Speaker(s)** |
| Processor | **Intel XScale 400 MHz** |
| Modem | **No** |
| Built-in devices | **Display** |

**Display**

| | |
|---|---|
| Display type | **3.8 in transreflective TFT** |

| | |
|---|---|
| Color support | **16-bit (65K colors)** |
| Max resolution | **320 x 240** |
| **Expansion/Connectivity** | |
| Connectivity device(s) included | **Docking cradle** |
| Expansion slot(s) total (free) | **1 x SD Memory Card** |
| Port(s) total (free) / connector type | **1 x IrDA, 1 x USB, 1 x serial RS-232, 1 x headphone, 1 x microphone, 1 x AC adapter** |
| Wireless connectivity | **Bluetooth, IrDA, Wi-Fi** |
| **Power** | |
| Battery installed (max) | **1 x Lithium Ion battery** |
| Power supply included | **Power adapter - external** |

**Appendix B: Table 4 - HP IPAQ 5450 Specification**

# APPENDIX C. HOW TO ENABLE SNMP SERVICE

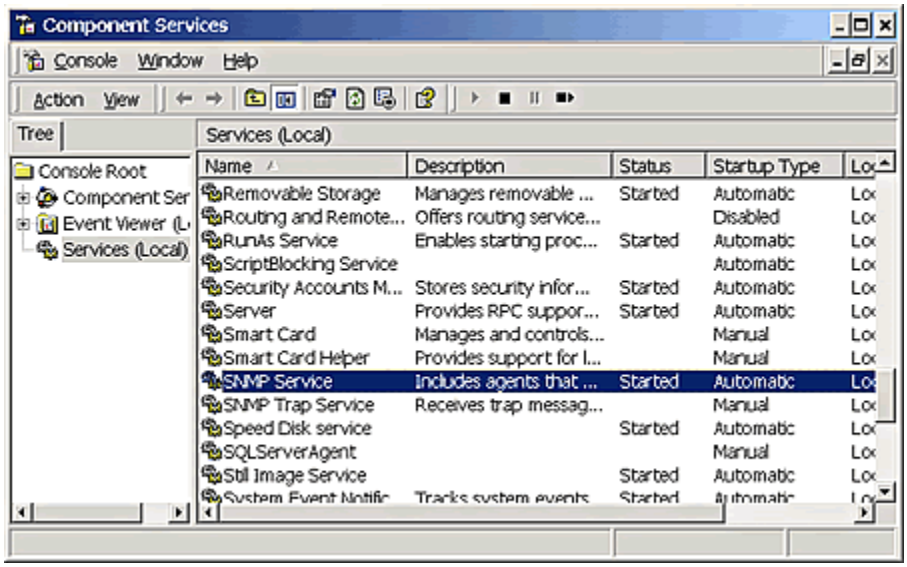
## A. ENABLE THE SNMP SERVICE

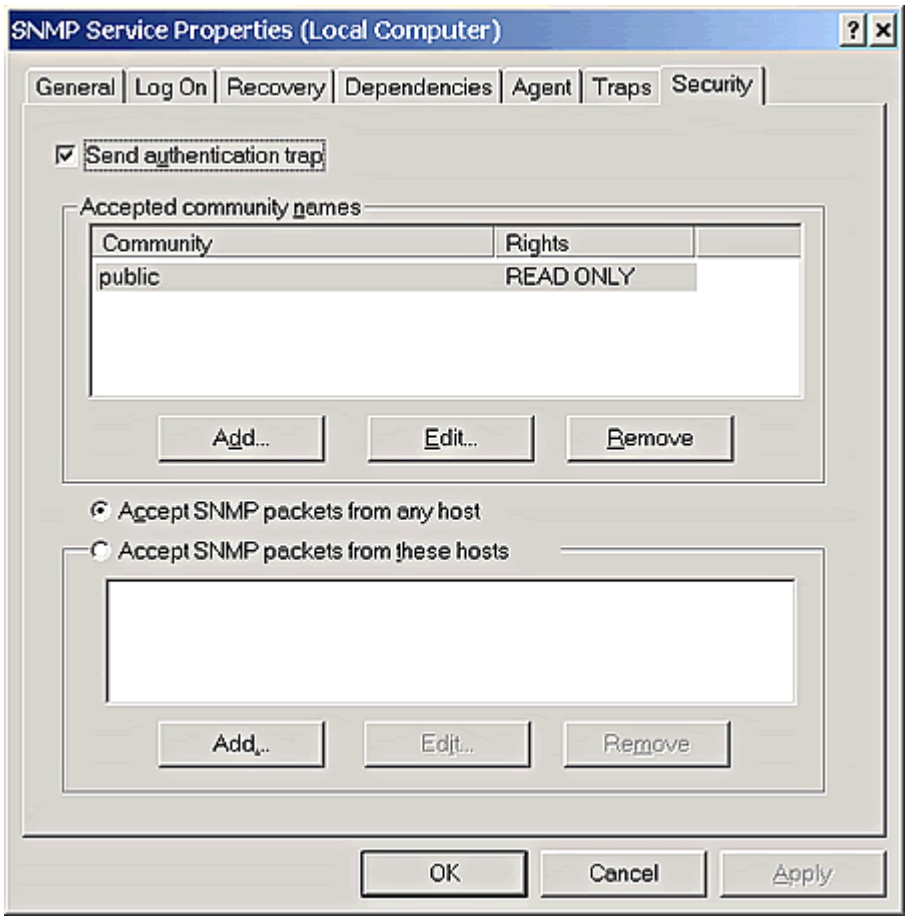
For Windows 2000/XP[17]

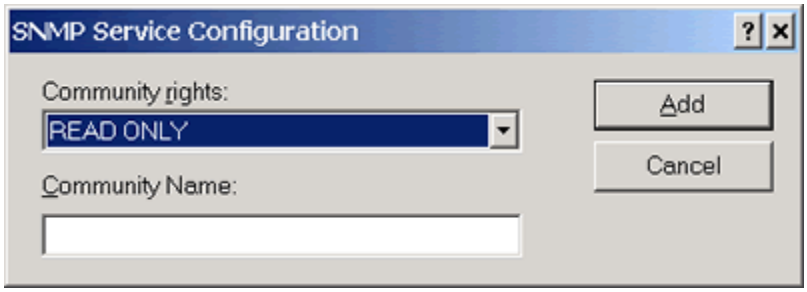
1. Login with Administrator status.
2. Open the Windows Components Wizard. To open the Windows Components wizard, click Start, point to Settings, click Control Panel, double-click Add/Remove Programs, and then click Add/Remove Windows Components.
3. In Components, click Management and Monitoring Tools (but do not select or clear its check box), and then click Details.
4. Select Simple Network Management Protocol check box, and click OK.
5. Click Next.

To ensure that the SNMP agent is installed go to the Control Panel and click on Administrative tools, then Component Services. Highlight Services in the left frame then scroll through the right frame to locate SNMP Service. However make sure that the SNMP agent is running before attempting to test it. This can be determined by looking at the status field. To ensure that the agents start up automatically right-click on SNMP Service then choose Properties. Under the General tab set the Startup type to automatic. Next, to set the community strings for this agent click on the Security tab then click on Add under the accepted Community Names. Here is the field for entering a community name. In this experiment, by disregarding the security scope, the default community string "public" will be used.

[17] Adapt from Windows 2000 help file


Component Services
Console Window Help
Action View
Tree
Services (Local)
Console Root
Component Ser
Event Viewer (L
Services (Local)
Name
Description
Status
Startup Type
Removable Storage
Manages removable ...
Started
Automatic
Routing and Remote...
Offers routing service...
Disabled
RunAs Service
Enables starting proc...
Started
Automatic
ScriptBlocking Service
Automatic
Security Accounts M...
Stores security infor...
Started
Automatic
Server
Provides RPC suppor...
Started
Automatic
Smart Card
Manages and controls...
Manual
Smart Card Helper
Provides support for l...
Manual
SNMP Service
Includes agents that ...
Started
Automatic
SNMP Trap Service
Receives trap messag...
Manual
Speed Disk service
Started
Automatic
SQLServerAgent
Manual
Stil Image Service
Started
Automatic


SNMP Service Properties (Local Computer)
General
Log On
Recovery
Dependencies
Agent
Traps
Security
Send authentication trap
Accepted community names
Community
Rights
public
READ ONLY
Add...
Edit...
Remove
Accept SNMP packets from any host
Accept SNMP packets from these hosts
Add...
Edit...
Remove
OK
Cancel
Apply




**Appendix C: Figure 1 - SNMP configuration in Windows 2000**

## B. SNMP CONFIGURATION

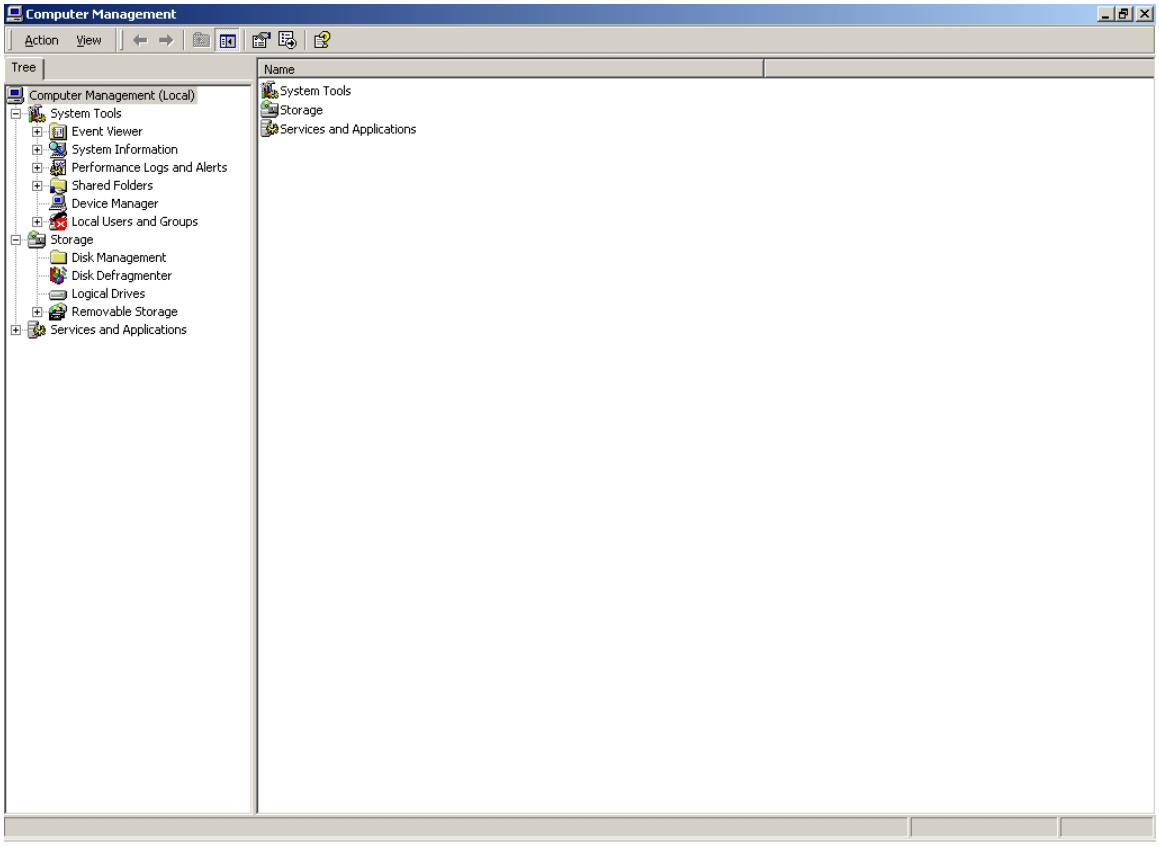
To configure agent properties for Windows 2000/XP[18]

1. Open Computer Management. [Figure 2 ]
2. In the console tree, click **Services**.
3. In the details pane, click **SNMP Service**
4. On the **Action** menu, click **Properties**. [Figure 3]
5. On the **Agent** tab, in **Contact**, type the name of the user or administrator for this computer. [Figure 4]
6. In **Location**, type the physical location of the computer or the contact.
7. Under **Service**, select the appropriate check boxes for this computer, and then click **OK**.

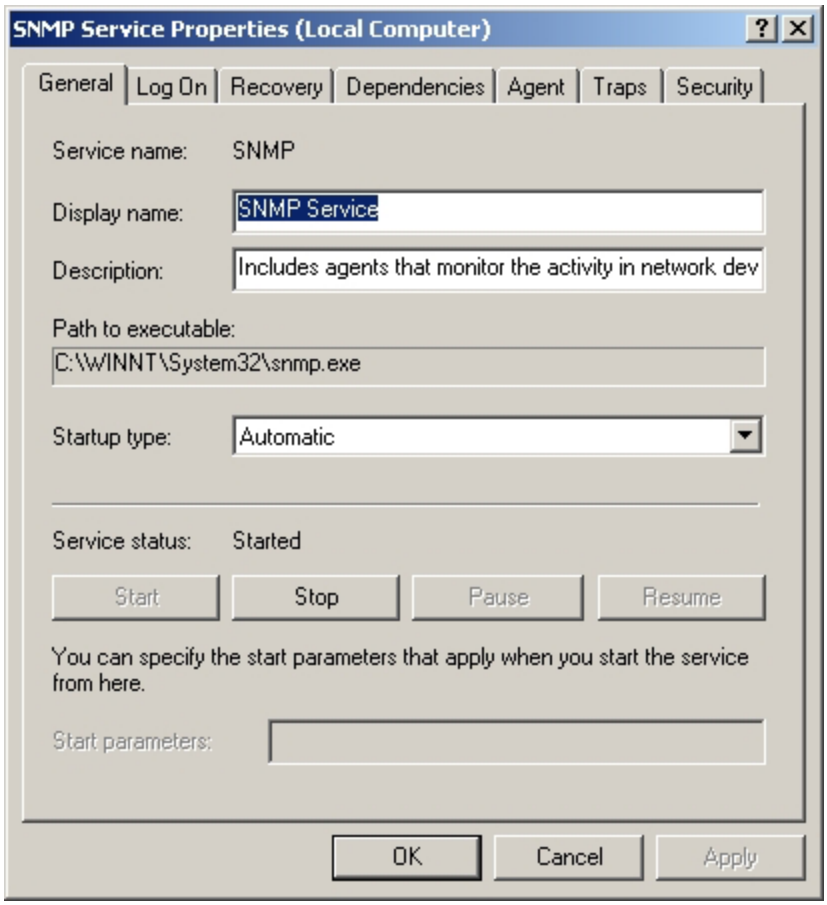
Notes

- To open Computer Management, click Start, point to Settings, and click Control Panel. Double-click Administrative Tools and then double-click Computer Management.
- If you change existing SNMP settings, your changes take effect immediately. If you are configuring SNMP for the first time, you must restart SNMP before these settings take effect.

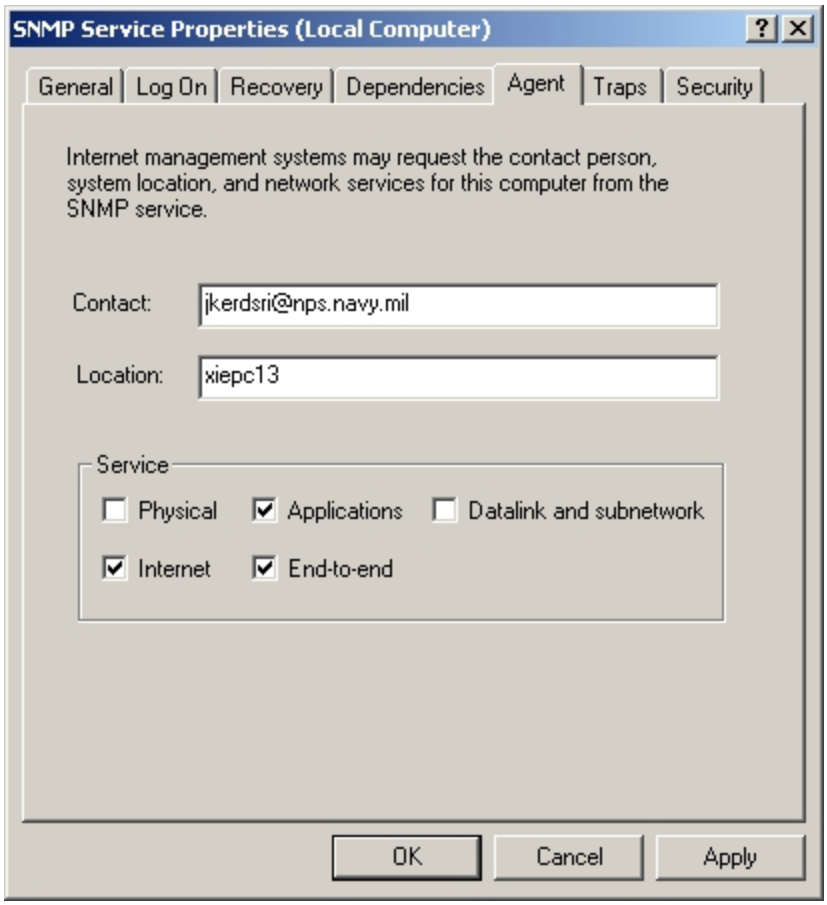
[18] adapted from Microsoft Windows 2000 Help File




**Appendix C: Figure 2 - The Computer Management Console**




**Appendix C: Figure 3 - SNMP Service Properties**



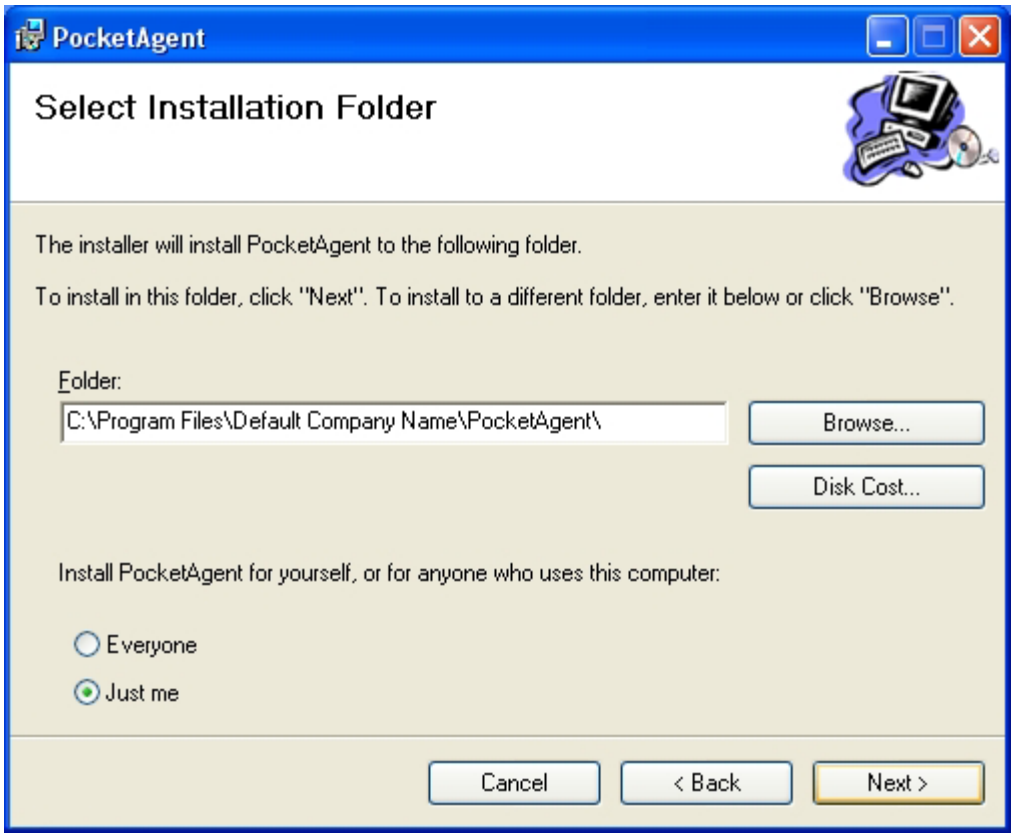

**Appendix C: Figure 4 - SNMP Service Properties - Agent Configuration**

# APPENDIX D. APPLICATION SETUP

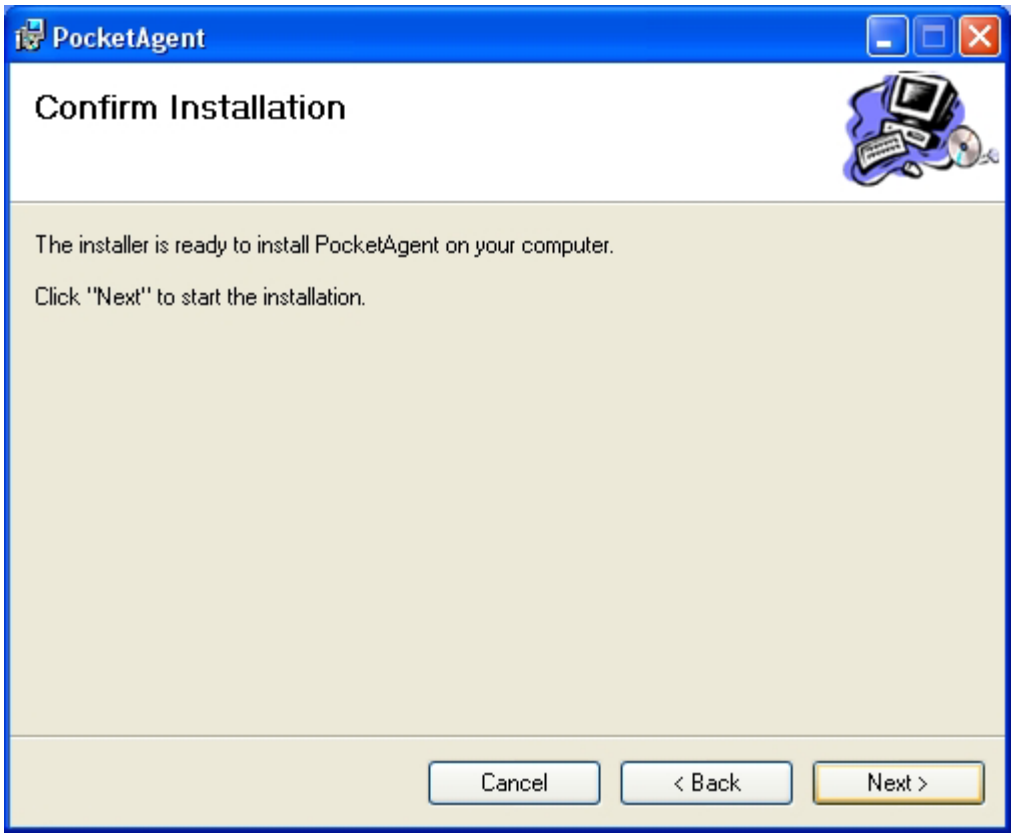
## A. POCKET AGENT PROTOTYPE SET UP

1. Double click the set up file in Debug folder then the set up Wizard will begin. [Figure 1]
2. Select the installation folder. The application will be created under the specify folder, the default folder is C:\Program Files\Default Company Name\PocketAgent\ [Figure 2]
3. Then click Next> [Figure 3]
4. Click Next>
5. The application is installed and ready to use. [Figure 4]
6. To run the program, direct to the installed folder and double click PocketAgent.exe file.

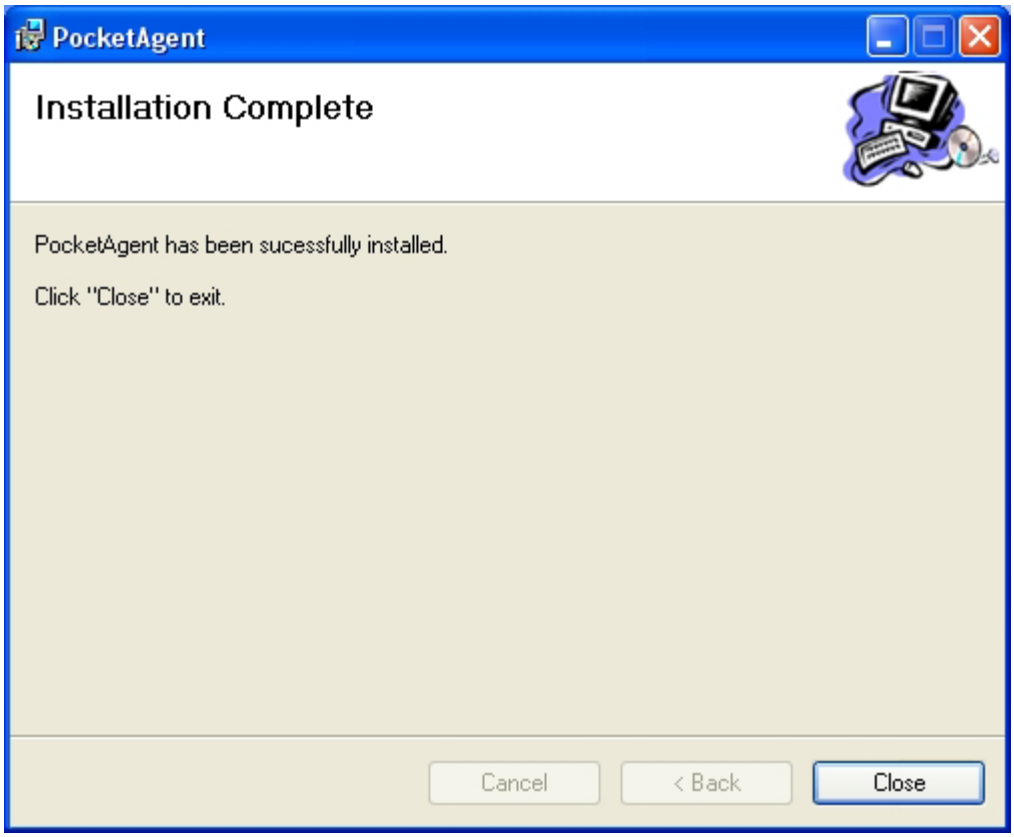

**Appendix D: Figure 1 – Set Up Wizard**




**Appendix D: Figure 2 – Select Installation Folder**




**Appendix D: Figure 3 – Confirm Installation**




**Appendix D: Figure 4 – Complete Installation**

# APPENDIX E. SNMP COMPONENT

## SNMP Component

The SNMP control is used to implement SNMP Management Applications and SNMP Agent Applications. (

The SNMP control implements a standard SNMP Version 1 Manager and/or Agent as specified in RFC 1157.

The control provides both encoding/decoding and transport capabilities, making the task of developing a custom SNMP agent or manager as simple as setting a few key properties and handling a few events. SNMP data, such as for instance SNMP object id-s (OID-s) are exchanged as text strings, thus further simplifying the task of handling them.

The control is activated/deactivated by first setting the LocalPort to 161 or 162, depending on whether you want to implement an SNMP agent or SNMP manager, and then setting the Active property. This property enables or disables sending or receiving. It operates completely asynchronously. Messages are sent to other agents or managers by using the Action property, and are recieved through events such as GetRequest, GetResponse, or Trap.

SNMP object ids, types, and values are provided in arrays such as ObjId, ObjType, and ObjValue, for both sent and received packets. ObjCount provides the number of elements in each of the arrays. Other packet information is provided through corresponding properties, such as Community, or RequestId, or similarly named parameters in events.

The control may behave as an SNMP agent or SNMP manager, depending on the value of the LocalPort property. If the control listens to port 161 it may act as an SNMP agent by responding to SNMP requests from SNMP managers, and sending traps through the Action property. If the LocalPort is set to 162, the control listens for SNMP traps, and may send requests to SNMP agents listening on port 161.

SNMP Traps are received through the Trap event, and may be sent through the Action property by specifying appropriate values in the various trap properties, such as TrapAgentAddress, TrapEnterprise, TrapGenericType, TrapSpecificType, and TrapTimeStamp.

This control requires a Winsock 1.1 compliant TCP/IP stack. This means that the Winsock stack installed in the system must have a version of at least 1.1. In particular, Windows 95, 98, and NT machines with Winsock 2.0 are fully supported.

## PROPERTIES

Action. An action code for the control.

Active. Enables or disables sending and receiving of SNMP packets.

Community. The community string used to authenticate SNMP packets.

ErrorIndex. Index of the first variable (object) that caused an error.

ErrorStatus. Status code for outgoing 'Get-Response' packets.

InBufferSize. The size in bytes of the incoming queue of the socket.

LocalPort. The UDP port in the local host where the SNMP control listens to.

ObjCount. Number of objects in the current request.

ObjId. Array of OIDs encoded as strings.

ObjType. Array of object types.

ObjValue. Array of object values.

OutBufferSize. The size in bytes of the outgoing queue of the socket.

RemoteHost. The address of the remote host. Domain names are resolved to IP addresses.

RemotePort. The UDP port where the remote SNMP agent is listening.

RequestId. The request-id to mark outgoing packets with.

TrapAgentAddress. The address of the object generating the trap.

TrapEnterprise. The type of the object generating the trap.

TrapGenericType. The generic type of the trap being sent.

TrapSpecificType. The specific type of the trap being sent.

TrapTimeStamp. Time passed since the agent was initialized (in hundredths of a second).

LocalHost. The name of the local host. When connected, the IP address of the interface through which the connection was made.

SocketHandle. The handle of the main socket used by the control.

WinsockInfo. Identifying information about the loaded Winsock stack.

WinsockLoaded. Loads and unloads Winsock on demand.

WinsockMaxDatagramSize. Size in bytes of the largest UDP datagram that can be sent or received.

WinsockMaxSockets. Maximum number of sockets available to a single process.

WinsockPath. The path to the Winsock DLL used.

WinsockStatus. The status of the Winsock stack.

## EVENTS

GetNextRequest. Fired when a GetNextRequest packet is received.

GetRequest. Fired when a GetRequest packet is received.

GetResponse. Fired when a GetResponse packet is received.

ReadyToSend. Fired when the control is ready to send data.

SetRequest. Fired when a SetRequest packet is received.

Trap. Fired when a SNMP trap packet is received.

Error. Information about errors during data delivery.

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF REFERENCES


[01] Michael Morrison, Using Pocket PC 2002, Que Publishing, June 2002.

[02] YoramCohen,SNMP,http://www2.rad.com/networks/1995/snmp/snmp.htm, accessed on 20 September 2002.

[03] http://www.airlinx.com/index.cfm/id/1-11.htm , accessed on 1 February 2003.

[04] William Stalling, SNMP, SNMPv2, SNMPv3, and RMON1 and 2, Addison-Wesley, 1999.

[05] Network Management in a Wireless Environment, white paper, Microsoft - http://www.symbol.com/products/whitepapers/whitepapers_network_mgmt_in_wi.html , accessed on 20 February 2003.

[06] Simple Network Management Protocol (SNMP), Cisco System Inc. - http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/snmp.htm, accessed on 20 February 2003.

[07] Allen Jones, C# for Java Developers, Microsoft press 2003.

[08] Bruce E. Krell PH.D., Pocket PC Developer's Guide, McGraw-Hill 2002.

[09] Microsoft Corporation White paper, writing your own MIB for Microsoft Windows CE 3.0, July 2002.

[10] http://computers.cnet.com/hardware/0-2709830-417-20886891.html?tag=dir , accessed on 3 March 2003.

THIS PAGE INTENTIONALLY LEFT BLANK

# INITIAL DISTRIBUTION LIST

1. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

2. Dr. Ted Lewis
   Department of Computer Science
   Naval Postgraduate School
   Monterey, CA

3. Dr. Geoffrey Xie
   Department of Computer Science
   Naval Postgraduate School
   Monterey, CA

4. Dr. Gurminder Singh
   Department of Computer Science
   Naval Postgraduate School
   Monterey, CA

5. LT Jiradett Kerdsri
   Supreme Headquarter
   Bangkok, Thailand

THIS PAGE INTENTIONALLY LEFT BLANK